CONTENTS

SUMMARY 1 1/A9

INTRODUCTION 1 1/A9

OVERVIEW OF ORACLS 2 1/A10

PRIMARY SUBROUTINES FOR INPUT-OUTPUT 5 1/A13
  Subroutine RDTITL[1] 5 1/A13
    (Input Hollerith data; define COMMON block data)
  Subroutine LNCNT 6 1/A14
    (Accumulate line count; page output)
  Subroutine READ 7 1/B1
    (Input numerical data)
  Subroutine PRNT 8 1/B2
    (Output numerical data)

PRIMARY SUBROUTINES FOR VECTOR-MATRIX OPERATIONS 10 1/B4
  Subroutine EQUATE 10 1/B4
    (Equating matrices, $A = B$)
  Subroutine TRANP 11 1/B5
    (Matrix transpose, A')
  Subroutine SCALE 12 1/B6
    (Scalar multiplication)
  Subroutine UNITY 13 1/B7
    (Generate identity matrix)
  Subroutine NULL 14 1/B8
    (Generate null matrix)
  Subroutine TRCE 15 1/B9
    (Trace of matrix)
  Subroutine ADD 16 1/B10
    (Matrix addition)
  Subroutine SUBT 17 1/B11
    (Matrix subtraction)
  Subroutine MULT 18 1/B12
    (Matrix multiplication)
  Subroutine MAXEL 19 1/B13
    (Maximum of absolute values of matrix elements)
  Subroutine NORMS 20 1/B14
    (Compute ℓ_1, ℓ_2, and ℓ_∞ matrix norms)
  Subroutine JUXTC 22 1/C2
    (Matrix juxtaposition by columns)
  Subroutine JUXTR 23 1/C3
    (Matrix juxtaposition by rows)

---

[1] Required of all user-provided driver programs.

PRIMARY SUBROUTINES FOR ANALYSIS OF CONSTANT LINEAR SYSTEMS 24 1/C4
Subroutine FACTOR 24 1/C4
(Factor nonnegative definite matrix)
Subroutine EIGEN 26 1/C6
(Eigenvalue-eigenvector computation)
Subroutine SYMPDS 28 1/C8
(Solve AX = B, A = A' > 0)
Subroutine GELIM 30 1/C10
(Solve AX = B, A nonsingular)
Subroutine SNVDEC 32 1/C12
(Singular value decomposition)
Subroutine SUM 35 1/D1
(Solve discrete Liapunov equation)
Subroutine BILIN 37 1/D3
(Solve continuous Liapunov equation)
Subroutine BARSTW 41 1/D7
(Solve general equation AX + XB = C)
Subroutine TESTSTA 43 1/D9
(Test matrix for relative stability)
Subroutine EXPSER 45 1/D11
(Matrix exponential by series method)
Subroutine EXPADE 47 1/D13
(Matrix exponential by Padé method)
Subroutine EXPINT 49 1/E1
(Matrix exponential and integral)
Subroutine VARANCE 51 1/E3
(Steady-state variance of linear system)
Subroutine CTROL 54 1/E6
(Controllability matrix)
Subroutine TRANSIT 57 1/E9
(Transient response of forced linear system)

PRIMARY SUBROUTINES FOR IMPLEMENTING LQG CONTROL LAW DESIGN 60 1/E12
Subroutine SAMPL 60 1/E12
(Evaluate sampled data regulator coefficients)
Subroutine PREFIL 64 1/F2
(Eliminate performance index cross-product terms)
Subroutine CSTAB 67 1/F5
(Stabilize continuous systems)
Subroutine DSTAB 70 1/F8
(Stabilize discrete systems)
Subroutine DISCREG 74 1/F12
(Digital transient quadratic regulator)
Subroutine CNTNREG 78 1/G2
(Continuous transient quadratic regulator)
Subroutine RICTNWT 84 1/G8
(Riccati solution by Newton's method)
Subroutine ASYMREG 88 1/G12
(Asymptotic quadratic regulator)
Subroutine ASYMFIL 94 2/A7
(Asymptotic Kalman-Bucy filter)

Subroutine EXPMDFL 100 2/A13
(Explicit model following)
Subroutine IMPMDFL 108 2/B7
(Implicit model following)

SUPPORTING SUBROUTINES 115 2/B14
Subroutine READ1 . . . . 115 2/B14
(Read single matrix)
Subroutine BALANC 116 2/C1
(Supports eigenvalue/eigenvector computation)
Subroutine ELMHES 117 2/C2
(Supports eigenvalue/eigenvector computation)
Subroutine HQR 118 2/C3
(Supports eigenvalue/eigenvector computation)
Subroutine INVIT 120 2/C5
(Supports eigenvalue/eigenvector computation)
Subroutine ELMBAK 122 2/C7
(Supports eigenvalue/eigenvector computation)
Subroutine BALBAK 123 2/C8
(Supports eigenvalue/eigenvector computation)
Subroutine DETFAC 124 2/C9
(LU factorization of real square matrix)
Subroutine AXPXB 126 2/C11
(Solve AX + XB = C)
Subroutine SHRSLV 129 2/C14
(Supports BARSTW)
Subroutine ATXPXA 130 2/D1
(Solve A'X + XA = C, C = C')
Subroutine SYMSLV 132 2/D3
(Supports BARSTW)
Subroutine HSHLDR 133 2/D4
(Supports BARSTW)
Subroutine BCKMLT 134 2/D5
(Supports BARSTW)
Subroutine SCHUR 135 2/D6
(Supports BARSTW)
Subroutine SYSSLV 137 2/D8
(Supports BARSTW)
Subroutine GAUSEL 138 2/D9
(Supports EXPADE)

EXAMPLE COMPUTATIONS 139 2/D10
Example 1 - Optimal Transient Regulator 140 2/D11
Example 2 - Optimal Sampled-Data Regulator 152 2/F5
Example 3 - Model Following 163 2/G11
Example 4 - Kalman-Bucy Filter 183 3/C11

CONCLUDING REMARKS 190 3/D9

REFERENCES 191 3/D10

MAY 1 0 1978

NAS 1.60: 1106

COMPLETED

NASA Technical Paper 1106

ORACLS - A System for Linear-Quadratic-Gaussian Control Law Design

Ernest S. Armstrong

APRIL 1978

NASA

199

NAS 1.60: 1106

NASA Technical Paper 1106

ORACLS - A System for Linear-Quadratic-Gaussian Control Law Design

Ernest S. Armstrong
Langley Research Center
Hampton, Virginia

National Aeronautics
and Space Administration

**Scientific and Technical
Information Office**

1978

CONTENTS

SUMMARY 1

INTRODUCTION 1

OVERVIEW OF ORACLS 2

PRIMARY SUBROUTINES FOR INPUT-OUTPUT 5
  Subroutine RDTITL[1] 5
    (Input Hollerith data; define COMMON block data)
  Subroutine LNCNT 6
    (Accumulate line count; page output)
  Subroutine READ 7
    (Input numerical data)
  Subroutine PRNT 8
    (Output numerical data)

PRIMARY SUBROUTINES FOR VECTOR-MATRIX OPERATIONS 10
  Subroutine EQUATE 10
    (Equating matrices, $A = B$)
  Subroutine TRANP 11
    (Matrix transpose, A')
  Subroutine SCALE 12
    (Scalar multiplication)
  Subroutine UNITY 13
    (Generate identity matrix)
  Subroutine NULL 14
    (Generate null matrix)
  Subroutine TRCE 15
    (Trace of matrix)
  Subroutine ADD 16
    (Matrix addition)
  Subroutine SUBT 17
    (Matrix subtraction)
  Subroutine MULT 18
    (Matrix multiplication)
  Subroutine MAXEL 19
    (Maximum of absolute values of matrix elements)
  Subroutine NORMS 20
    (Compute ℓ_1, ℓ_2, and ℓ_∞ matrix norms)
  Subroutine JUXTC 22
    (Matrix juxtaposition by columns)
  Subroutine JUXTR 23
    (Matrix juxtaposition by rows)

---

[1] Required of all user-provided driver programs.

PRIMARY SUBROUTINES FOR ANALYSIS OF CONSTANT LINEAR SYSTEMS 24
Subroutine FACTOR 24
(Factor nonnegative definite matrix)
Subroutine EIGEN 26
(Eigenvalue-eigenvector computation)
Subroutine SYMPDS 28
(Solve $AX = B$, $A = A' > 0$)
Subroutine GELIM 30
(Solve $AX = B$, A nonsingular)
Subroutine SNVDEC 32
(Singular value decomposition)
Subroutine SUM 35
(Solve discrete Liapunov equation)
Subroutine BILIN 37
(Solve continuous Liapunov equation)
Subroutine BARSTW 41
(Solve general equation $AX + XB = C$)
Subroutine TESTSTA 43
(Test matrix for relative stability)
Subroutine EXPSER 45
(Matrix exponential by series method)
Subroutine EXPADE 47
(Matrix exponential by Padé method)
Subroutine EXPINT 49
(Matrix exponential and integral)
Subroutine VARANCE 51
(Steady-state variance of linear system)
Subroutine CTROL 54
(Controllability matrix)
Subroutine TRANSIT 57
(Transient response of forced linear system)

PRIMARY SUBROUTINES FOR IMPLEMENTING LQG CONTROL LAW DESIGN 60
Subroutine SAMPL 60
(Evaluate sampled data regulator coefficients)
Subroutine PREFIL 64
(Eliminate performance index cross-product terms)
Subroutine CSTAB 67
(Stabilize continuous systems)
Subroutine DSTAB 70
(Stabilize discrete systems)
Subroutine DISCREG 74
(Digital transient quadratic regulator)
Subroutine CNTNREG 78
(Continuous transient quadratic regulator)
Subroutine RICTNWT 84
(Riccati solution by Newton's method)
Subroutine ASYMREG 88
(Asymptotic quadratic regulator)
Subroutine ASYMFIL 94
(Asymptotic Kalman-Bucy filter)

Subroutine EXPMDFL 100
(Explicit model following)
Subroutine IMPMDFL 108
(Implicit model following)

SUPPORTING SUBROUTINES 115
Subroutine READ1 115
(Read single matrix)
Subroutine BALANC 116
(Supports eigenvalue/eigenvector computation)
Subroutine ELMHES 117
(Supports eigenvalue/eigenvector computation)
Subroutine HQR 118
(Supports eigenvalue/eigenvector computation)
Subroutine INVIT 120
(Supports eigenvalue/eigenvector computation)
Subroutine ELMBAK 122
(Supports eigenvalue/eigenvector computation)
Subroutine BALBAK 123
(Supports eigenvalue/eigenvector computation)
Subroutine DETFAC 124
(LU factorization of real square matrix)
Subroutine AXPXB 126
(Solve AX + XB = C)
Subroutine SHRSLV 129
(Supports BARSTW)
Subroutine ATXPXA 130
(Solve A'X + XA = C, C = C')
Subroutine SYMSLV 132
(Supports BARSTW)
Subroutine HSHLDR 133
(Supports BARSTW)
Subroutine BCKMLT 134
(Supports BARSTW)
Subroutine SCHUR 135
(Supports BARSTW)
Subroutine SYSSLV 137
(Supports BARSTW)
Subroutine GAUSEL 138
(Supports EXPADE)

EXAMPLE COMPUTATIONS 139
Example 1 - Optimal Transient Regulator 140
Example 2 - Optimal Sampled-Data Regulator 152
Example 3 - Model Following 163
Example 4 - Kalman-Bucy Filter 183

CONCLUDING REMARKS 190

REFERENCES 191

SUMMARY

A modern control theory design package (ORACLS) for constructing controllers and optimal filters for systems modeled by linear time-invariant differential or difference equations is described. The digital FORTRAN-coded ORACLS system represents an application of some of today's best numerical linear-algebra procedures to implement the linear-quadratic-Gaussian (LQG) methodology of modern control theory. Included are algorithms for computing eigensystems of real matrices, the relative stability of a matrix, factored forms for nonnegative definite matrices, the solutions and least squares approximations to the solutions of certain linear matrix algebraic equations, the controllability properties of a linear time-invariant system, and the steady-state covariance matrix of an open-loop stable system forced by white noise. These subroutines are applied to implement the various techniques of the LQG methodology. Subroutines are provided for solving both the continuous and discrete optimal linear regulator problems with noise-free measurements and the sampled-data optimal linear regulator problem. For measurement noise, duality theory and the optimal regulator algorithms are used to solve the continuous and discrete Kalman-Bucy filter problems. Subroutines are also included which give control laws causing the output of a system to track the output of a prescribed model. Finally, numerical examples are presented to illustrate the capability of the ORACLS system.

INTRODUCTION

Optimal linear quadratic regulator theory, currently referred to as the linear-quadratic-Gaussian (LQG) problem (ref. 1), has become the most widely accepted method of determining optimal control policy. Within this class of optimal control problems, the infinite-duration time-invariant version, which leads to constant-gain feedback control laws and constant Kalman-Bucy filter gains for reconstruction of the system state, has received the bulk of the attention because of its tractability and potential ease of implementation. The theory is particularly attractive to the control system designer because it provides a rigorous tool for dealing with multi-input and multi-output dynamical systems in both continuous and discrete form.

During the same period of time that the LQG methodology was being developed, there also appeared in the field of numerical linear algebra a variety of new and more efficient methods for analyzing the types of equations which occur in the time-invariant formulation of the LQG problem (ref. 2). The best of these numerical methods (in the opinion of the author) have been incorporated into a modern efficient digital computer package which provides solutions to time-invariant continuous or discrete LQG problems with equal ease. The package is entitled Optimal Regulator Algorithms for the Control of Linear Systems (ORACLS).

An overview of the features of the ORACLS system is presented in the next section. Following sections give a detailed description of the package contents along with numerical examples to illustrate the use of ORACLS to solve selected LQG problems. Software for the ORACLS system may be obtained from the centralized facility COSMIC located at the Computer Software Management and Information Center, Suite 112, Barrow Hall, University of Georgia, Athens, GA 30602, by requesting program LAR-12313.

OVERVIEW OF ORACLS

The ORACLS programing system is a collection of FORTRAN-coded subroutines to formulate, manipulate, and solve various LQG design problems. In order to apply ORACLS, the user is required to provide an executive (driver) program which inputs the problem coefficients, formulates and selects the system subroutines to be used to solve a particular optimal control problem, and outputs desired information. ORACLS is constructed to allow the user considerable flexibility at each operational state. This flexibility is accomplished by providing primary subroutines at four levels: input-output, basic vector-matrix operations, analysis of linear time-invariant systems, and control synthesis based on LQG methodology. ORACLS provides a means of controlling program size by employing dynamic (vector) data storage. Except for certain subroutines obtained from other sources, data arrays in all ORACLS subroutines are treated as packed one-dimensional arrays which can easily be passed between subroutines without a maximum array size parameter appearing as an argument of the calling sequence. This dynamic storage capability allows program size to be specified and controlled through the user's driver program. In addition, ORACLS only loads those programs from the library which are called by the executive program, making the total machine requirements very flexible. As a result, ORACLS can be made to execute efficiently on a wide variety of computing machinery.

The input-output category of ORACLS has subroutines for inputing (subroutine READ) and outputing (PRNT) numerical matrices. Additional subroutines allow for printing header information (RDTITL) and accumulation of output line-count information (LNCNT).

The next category has subroutines for the basic vector-matrix operations of equation (EQUATE), addition (ADD), subtraction (SUBT), and multiplication (MULT). It also contains routines for scaling (SCALE), juxtaposition (JUXTC and JUXTR), and construction of matrix norms (MAXEL and NORMS), trace (TRCE), transpose (TRANP), and null and identity matrices (NULL and UNITY).

The analysis category provides special and general purpose algorithms for computing (1) eigenvalues and eigenvectors of real matrices (EIGEN) by using the QR algorithm (ref. 2), (2) the relative stability of a given matrix (TESTSTA), (3) matrix factorization (FACTOR), (4) the solution of linear constant-coefficient vector-matrix algebraic equations (SYMPDS, GELIM, and SNVDEC), (5) the controllability properties of a linear time-invariant system (CTROL), (6) the steady-state covariance matrix of an open-loop stable system forced by white noise (VARANCE), and (7) the transient response of continuous linear time-invariant systems (TRANSIT).

Algorithms are provided for the solution of real matrix equations. In the equation,

$$AX = B \tag{1}$$

with A positive definite, the Cholesky decomposition method (ref. 2) is applied (SYMPDS). Gaussian elimination (LU factorization from ref. 2) is used for the general case of nonsingular A (GELIM). For rectangular or singular matrices A, singular-value decomposition procedures found in subroutine SNVDEC (ref. 2) can be used to find a solution of equation (1) in the least squares sense, to compute the pseudoinverse of A, and to find an orthonormal basis for the range space of A. A maximal rank matrix factorization for a positive semidefinite matrix can be obtained from FACTOR.

For solution of the matrix equation,

$$X = AXB + C \tag{2}$$

the contraction mapping principle (ref. 3) is applied when A and B are asymptotically stable in the discrete sense (SUM). Equation (2) is used when solving the discrete steady-state Riccati equation by Newton's method and also in the computation of the steady-state covariance matrix for a linear asymptotically stable discrete system forced by white noise (ref. 4).

Two subroutines are included for solving the matrix equation,

$$AX + XB = C \tag{3}$$

For A and B admitting a unique solution X, the method of Bartels and Stewart (ref. 5) is used (BARSTW). For A and B asymptotically stable in the continuous sense, a subroutine implementing the bilinear transformation approach (ref. 6) is also included (BILIN). Equation (3) is used in solving the steady-state continuous matrix Riccati equation by Newton's method, in finding the covariance statistics for continuous time-invariant systems forced by white noise, and in gain computation for observer theory (ref. 7).

A subroutine (CTROL) is provided which computes the controllability matrix for a linear time-invariant dynamical system and, if this matrix is found to be rank deficient, also computes the system's controllability canonical form (ref. 4) by application of the singular-value decomposition algorithm. Through this subroutine, the user may examine the stabilizability and detectability conditions implicit in the infinite-duration LQG methodology and, indirectly, compute minimal order state space realizations for transfer matrices.

Finally, the analysis category of ORACLS includes subroutines (EXPSER, EXPADE, and EXPINT) for computing

$$e^{At} \text{ and } \int_0^T e^{At}B\,dt$$

These expressions are used in computing the transient response of linear time-invariant dynamical systems (TRANSIT).

Subroutines are presented in the control law design part of ORACLS to implement some of the more common techniques of time-invariant LQG methodology. For the finite-duration optimal linear regulator problem with noise-free measurements, continuous dynamics, and integral performance index, a subroutine (CNTNREG) is provided to implement the negative exponential method for finding both the transient and steady-state solutions to the matrix Riccati equation (ref. 8). For the discrete version of this problem, the method of backward differencing is applied to find the transient and steady-state solutions to the discrete Riccati equation (DISCREG). For the infinite-duration optimal linear regulator problem with noise-free measurements, a subroutine is also included to solve the steady-state Riccati equation by the Newton algorithms described by Kleinman (ref. 9) for continuous problems and by Hewer (ref. 10) for discrete problems (RICTNWT). The methods described by Armstrong (ref. 11) and Armstrong and Rublein (ref. 12) are used to compute a stabilizing gain to initialize the continuous and discrete Newton iterations (CSTAB and DSTAB). A subroutine (PREFIL) is available for finding the prefilter gain to eliminate control-state cross-product terms in the quadratic performance index and another (SAMPL) computes the weighting matrices for the sampled-data optimal linear regulator problem (ref. 13). For cases with measurement noise, duality theory (ref. 4) and the foregoing optimal regulator algorithms are used to produce solutions to the continuous and discrete Kalman-Bucy filter problems (ASYMFIL). Finally, subroutines are included to implement the continuous (ref. 14) and discrete (ref. 15) forms of explicit (model in the system) and implicit (model in the performance index) model-following theory (EXPMDFL and IMPMDFL). These subroutines generate linear control laws which cause the output of a time-invariant dynamical system to track the output of a prescribed model.

In addition to the foregoing 43 primary purpose subroutines, ORACLS contains another 17 supporting subroutines used predominately by the algebraic equation and eigenvalue algorithms. All subroutines of the ORACLS package are (1) original codes by the author or (2) direct or author-modified copies of programs obtained from the VASP program (ref. 16), the software library of the Analysis and Computation Division at the Langley Research Center (LaRC), or the numerical analysis literature. The programs coded by the author and those obtained from the VASP program employ the dynamic storage capability in which data arrays are treated as packed one-dimensional arrays. Other subroutines were left in their original format.

A detailed description of each ORACLS subroutine can be found in the following sections. In the last section, numerical examples are presented to illustrate the capability of ORACLS in both digital and continuous LQG controller design and, additionally, to demonstrate the construction of typical executive programs.

PRIMARY SUBROUTINES FOR INPUT-OUTPUT

Subroutine RDTITL

Description: The purpose of RDTITL is to read a single card of Hollerith input which is loaded into the array TITLE of COMMON block LINES of RDTITL and automatically printed at the top of each page of output through the subroutine LNCNT. The Hollerith input is typically used to define, for future reference, the problem being solved by ORACLS.

Subroutine RDTITL also serves to define certain data blocks important to other subroutines within ORACLS, such as information for COMMON blocks LINES and FORM discussed in the description of LNCNT and PRNT, respectively. The subroutines BARSTW and SNVDEC require input parameters (EPSA, EPSB, and IAC) designating the accuracy to which solutions are to be obtained. When these programs are used internally to other subroutines, the accuracy parameters are set to values EPSAM, EPSBM, and IACM defined by DATA statements and contained in COMMON block TOL of RDTITL. Also, convergence parameters for terminating the recursive computations in subroutines SUM, EXPSER, EXPINT, SAMPL, DISCREG, CNTNREG, and RICTNWT are internally set. Parameters SUMCV (for SUM), RICTCV (for DISCREG, CNTNREG, and RICTNWT), MAXSUM (for SUM), and SERCV (for EXPSER, EXPINT, and SAMPL) are defined by DATA statements and contained in the COMMON block CONV. The user should specify the parameters of all COMMON blocks of RDTITL on the basis of his particular computing installation and problem to be solved.

Subroutine RDTITL must be a part of every executive program provided by the user of ORACLS. If not, extraneous data appearing in the array TITLE will be printed at the top of each page of output and other COMMON block data will be undefined.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL RDTITL

Input arguments: None

Output arguments: None

COMMON blocks: LINES, FORM, TOL, CONV

Error messages: None

Field length: 32 octal words (26 decimal)

Subroutine employed by RDTITL: LNCNT

Subroutines employing RDTITL: None

Comments: None

Subroutine LNCNT

Description: The purpose of LNCNT is to keep track of the number of lines printed and automatically paginate the output. Page length is controlled by the variable NLP set in the COMMON block LINES of subroutine RDTITL to 44. The variable NLP is an installation-dependent variable and may be changed as necessary. Subroutine LNCNT provides one line of print at the top of each page. This line contains 100 characters of which the first 80 are available for the programer's use and may be loaded by use of the subroutine RDTITL. The remainder contain "ORACLS PROGRAM." The 100 characters are contained in the array TITLE within RDTITL.

Source of software: VASP (ref. 16) with modifications by Ernest S. Armstrong, LaRC

Calling sequence: CALL LNCNT(N)

Input argument:

N	Number of lines to be printed

Output arguments: None

COMMON block: LINES

Error messages: None

Field length: 27 octal words (23 decimal)

Subroutines employed by LNCNT: None

Subroutines employing LNCNT: RDTITL, PRNT, EQUATE, TRANP, SCALE, UNITY, TRCE, ADD, SUBT, MULT, JUXTC, JUXTR, FACTOR, SUM, BILIN, BARSTW, TESTSTA, EXPSER, EXPINT, VARANCE, CTROL, TRANSIT, SAMPL, PREFIL, CSTAB, DSTAB, DISCREG, CNTNREG, RICTNWT, ASYMREG, ASYMFIL, EXPMDFL, IMPMDFL, READ1

Comments: Subroutine LNCNT is completely internal to the ORACLS subroutines and the user need not refer to it unless he has a WRITE statement of his own. In that case, the user should put the statement CALL LNCNT(N) before each WRITE statement.

Subroutine READ

Description: The purpose of READ is to read from one to five matrices from cards along with their names and dimensions and print the same information. For each matrix, a header card is first read containing a four-character title followed by two integers giving the row and column size of the matrix using format (4H,4X,2I4). Then, the matrix data are read by rows using subroutine READ1 (each row of the matrix starting on a new card) using format (8F10.2). Each matrix is then automatically printed using subroutine PRNT called from READ1 and packed by columns into one-dimensional arrays with the same names.

Source of software: VASP (ref. 16) with modifications by Ernest S. Armstrong, LaRC

Calling sequence: CALL READ(I,A,NA,B,NB,C,NC,D,ND,E,NE)

Input arguments:

I	Integer from 1 to 5 indicating the number of matrices to be read. For I < 5, the entries past I in the argument list may be omitted.
A, B, C, D, E	Input matrices
NA, NB, NC, ND, NE	Two-dimensional vectors giving the number of rows and columns of the respective matrices and input through the header card; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A If 0 is loaded for the row and column size, then the current matrix storage is unchanged, no data cards are read, and the previously stored matrix is printed.

Output arguments: None

COMMON blocks: None

Error message: None directly from READ

Field length: 155 octal words (109 decimal)

Subroutine employed by READ: READ1

Subroutines employing READ: None

Comments: None

Subroutine PRNT

Description: The purpose of PRNT is to print a single matrix with or without a descriptive heading either on the same page or on a new page. The descriptive heading, if desired, is printed before each matrix and is of the form "NAM MATRIX NA(1) ROWS NA(2) COLUMNS." The matrix is next printed by rows using format (1P7E16.7) for the first line and (3X1P7E16.7) for all subsequent lines. Format for the printing is stored in COMMON block FORM of RDTITL.

Source of software: VASP (ref. 16) with modifications by Ernest S. Armstrong, LaRC

Calling sequence: CALL PRNT(A,NA,NAM,IOP)

Input arguments:

A	Matrix packed by columns in a one-dimensional array
NA	Two-dimensional vector giving the number of rows and columns of the matrix A: NA(1) = Number of rows of A NA(2) = Number of columns of A
NAM	Hollerith characters giving matrix name. Generally, NAM should contain 4 Hollerith characters and be written in the argument list as 4HXXXX. Alternatively, if 0 is inserted in the argument list for NAM, a blank name is printed.
IOP	Scalar print control parameter: 1 Print heading and matrix on same page. 2 Print heading and matrix after skipping to next page. 3 Print only matrix with no heading on same page. 4 Print only matrix with no heading after skipping to next page.

Output arguments: None

COMMON blocks: FORM, LINES

Error message: If NA(1) × NA(2) < 1 or NA(1) < 1, the message "ERROR IN PRNT MATRIX ______ HAS NA = ______ ______" is printed, and the program is returned to the calling point.

Field length: 262 octal words (178 digital)

Subroutine employed by PRNT: LNCNT

Subroutines employing PRNT: FACTOR, SUM, BILIN, BARSTW, TESTSTA, EXPSER, EXPINT, VARANCE, CTROL, TRANSIT, SAMPL, PREFIL, CSTAB, DSTAB, DISCREG, CNTNREG, RICTNWT, ASYMREG, ASYMFIL, EXPMDFL, IMPMDFL, READ1

Comments: None

PRIMARY SUBROUTINES FOR VECTOR-MATRIX OPERATIONS

Subroutine EQUATE

Description: The purpose of EQUATE is to store a matrix in an alternate computer location.

Source of software: VASP (ref. 16) with modifications by Ernest S. Armstrong, LaRC

Calling sequence: CALL EQUATE(A,NA,B,NB)

Input arguments:

A	Matrix packed by columns in a one-dimensional array; not destroyed upon return
NA	Two-dimensional vector giving the number of rows and columns of A: NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return

Output arguments:

B	Matrix packed by columns in a one-dimensional array. Upon normal return, B = A.
NB	Two-dimensional vector: NB = NA upon normal return

COMMON blocks: None

Error message: If NA(1) × NA(2) < 1 or NA(1) < 1, the message "DIMENSION ERROR IN EQUATE NA = ______ ______" is printed, and the program is returned to the calling point.

Field length: 46 octal words (38 decimal)

Subroutine employed by EQUATE: LNCNT

Subroutines employing EQUATE: FACTOR, SUM, BILIN, BARSTW, TESTSTA, EXPSER, EXPINT, VARANCE, CTROL, TRANSIT, SAMPL, PREFIL, CSTAB, DSTAB, DISCREG, CNTNREG, RICTNWT, ASYMREG, ASYMFIL, EXPMDFL, IMPMDFL

Comments: None

Subroutine TRANP

Description: The purpose of TRANP is to compute the transpose A' of a given matrix A.

Source of software: VASP (ref. 16) with modifications by Ernest S. Armstrong, LaRC

Calling sequence: CALL TRANP(A,NA,B,NB)

Input arguments:

A	Matrix packed by columns in a one-dimensional array; not destroyed upon return
NA	Two-dimensional vector giving the number of rows and columns of A: NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return

Output arguments:

B	Matrix packed by columns in a one-dimensional array. Upon normal return, B = A'.
NB	Two-dimensional vector: upon normal return, NB(1) = NA(2) NB(2) = NA(1)

COMMON blocks: None

Error message: If NA(1) × NA(2) < 1 or NA(1) < 1, the message "DIMENSION ERROR IN TRANP NA = ______ ______" is printed, and the program is returned to the calling point.

Field length: 106 octal words (70 decimal)

Subroutine employed by TRANP: LNCNT

Subroutines employing TRANP: FACTOR, SUM, BILIN, BARSTW, VARANCE, CTROL, TRANSIT, SAMPL, PREFIL, CSTAB, DSTAB, DISCREG, CNTNREG, RICTNWT, ASYMREG, ASYMFIL, EXPMDFL, IMPMDFL

Comments: None

Subroutine SCALE

Description: The purpose of SCALE is to perform scalar multiplication on a given matrix.

Source of software: VASP (ref. 16) with modifications by Ernest S. Armstrong, LaRC

Calling sequence: CALL SCALE(A,NA,B,NB,S)

Input arguments:

A	Matrix packed by columns in one-dimensional array; not destroyed upon return
NA	Two-dimensional vector giving the number of rows and columns of A: NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return
S	Scalar

Output arguments:

B	Matrix packed by columns in a one-dimensional array. Upon normal return, B = SA.
NB	Two-dimensional vector: NB = NA upon normal return

COMMON blocks: None

Error message: If NA(1) × NA(2) < 1 or NA(1) < 1, the message "DIMENSION ERROR IN SCALE NA = ______ ______" is printed, and the program is returned to the calling point.

Field length: 47 octal words (39 decimal)

Subroutine employed by SCALE: LNCNT

Subroutines employing SCALE: FACTOR, BILIN, EXPSER, EXPINT, VARANCE, TRANSIT, SAMPL, PREFIL, CSTAB, DSTAB, CNTNREG, RICTNWT, ASYMREG, EXPMDFL, IMPMDFL

Comments: None

Subroutine UNITY

Description: The purpose of UNITY is to generate an identity matrix.

Source of software: VASP (ref. 16) with modifications by Ernest S. Armstrong, LaRC

Calling sequence: CALL UNITY(A,NA)

Input argument:

NA	Two-dimensional vector giving dimension of square identity matrix: NA(1) = NA(2) = Number of rows of A Not destroyed upon return

Output argument:

A	Matrix packed by columns in a one-dimensional array. Upon normal return, A = I where I is an identity matrix of order NA(1).

COMMON blocks: None

Error message: If NA(1) ≠ NA(2), the message "DIMENSION ERROR IN UNITY NA = ______ ______" is printed, and the program is returned to the calling point.

Field length: 61 octal words (49 decimal)

Subroutine employed by UNITY: LNCNT

Subroutines employing UNITY: BILIN, EXPSER, EXPINT, TRANSIT

Comments: None

Subroutine NULL

Description: The purpose of NULL is to generate a null matrix.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL NULL(A,NA)

Input argument:

NA	Two-dimensional vector giving the number of rows and columns of the desired null matrix: NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return

Output argument:

A	Matrix packed by columns in a one-dimensional array. Upon normal return, A = 0 where 0 is a null matrix of order NA(1) × NA(2).

COMMON blocks: None

Error message: If NA(1) × NA(2) < 1 or NA(1) < 1, the message "DIMENSION ERROR IN NULL NA = ______ ______" is printed, and the program is returned to the calling point.

Field length: 35 octal words (29 decimal)

Subroutine employed by NULL: LNCNT

Subroutines employing NULL: BARSTW, CNTNREG

Comments: None

Subroutine TRCE

Description: The purpose of TRCE is to compute the trace of a square matrix.

Source of software: VASP (ref. 16) with modifications by Ernest S. Armstrong, LaRC

Calling sequence: CALL TRCE(A,NA,TR)

Input arguments:

A	Matrix packed by columns in a one-dimensional array; not destroyed upon return
NA	Two-dimensional vector giving number of rows and columns of A: NA(1) = NA(2) = Order of A Not destroyed upon return

Output argument:

TR	Scalar. Upon normal return, TR = Trace of A.

COMMON blocks: None

Error message: If NA(1) ≠ NA(2), the message "TRACE REQUIRES SQUARE MATRIX NA = ______ ______" is printed, and the program is returned to the calling point.

Field length: 52 octal words (42 decimal)

Subroutine employed by TRCE: LNCNT

Subroutine employing TRCE: EXPSER

Comments: None

Subroutine ADD

Description: The purpose of ADD is to perform matrix addition $C = A + B$ for given matrices A and B.

Source of software: VASP (ref. 16) with modifications by Ernest S. Armstrong, LaRC

Calling sequence: CALL ADD(A,NA,B,NB,C,NC)

Input arguments:

A, B	Matrices packed by columns in one-dimensional arrays; not destroyed upon return
NA, NB	Two-dimensional vectors giving number of rows and columns of respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return

Output arguments:

C	Matrix packed by columns in a one-dimensional array. Upon normal return, $C = A + B$.
NC	Two-dimensional vector: upon normal return, NC(1) = NA(1) NC(2) = NA(2)

COMMON blocks: None

Error message: If either $NA(1) \neq NB(1)$, $NA(2) \neq NB(2)$, $NA(1) < 1$, or $NA(1) \times NA(2) < 1$, the message "DIMENSION ERROR IN ADD NA = ______ ______, NB = ______ ______" is printed, and the program is returned to the calling point.

Field length: 56 octal words (46 decimal)

Subroutine employed by ADD: LNCNT

Subroutines employing ADD: SUM, EXPSER, EXPINT, TRANSIT, SAMPL, DSTAB, DISCREG, CNTNREG, RICTNWT, ASYMREG, EXPMDFL, IMPMDFL

Comments: None

Subroutine SUBT

Description: The purpose of SUBT is to perform matrix subtraction C = A - B for given matrices A and B.

Source of software: VASP (ref. 16) with modifications by Ernest S. Armstrong, LaRC

Calling sequence: CALL SUBT(A,NA,B,NB,C,NC)

Input arguments:

A, B	Matrices packed by columns in one-dimensional arrays; not destroyed upon return
NA, NB	Two-dimensional vectors giving the number of rows and columns of respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return

Output arguments:

C	Matrix packed by columns in a one-dimensional array. Upon normal return, C = A - B.
NC	Two-dimensional vector: upon normal return, NC(1) = NA(1) NC(2) = NA(2)

COMMON blocks: None

Error message: If either NA(1) ≠ NB(1), NA(2) ≠ NB(2), NA(1) < 1, or NA(1) × NA(2) < 1, the message "DIMENSION ERROR IN SUBT NA = ______ ______, NB = ______ ______" is printed, and the program is returned to the calling point.

Field length: 56 octal words (46 digital)

Subroutine employed by SUBT: LNCNT

Subroutines employing SUBT: TRANSIT, PREFIL, CSTAB, DSTAB, DISCREG, CNTNREG, RICTNWT, ASYMREG, EXPMDFL, IMPMDFL

Comments: None

Subroutine MULT

Description: The purpose of MULT is to perform matrix multiplication $C = AB$ for given matrices A and B.

Source of software: VASP (ref. 16) with modifications by Ernest S. Armstrong, LaRC

Calling sequence: `CALL MULT(A,NA,B,NB,C,NC)`

Input arguments:

A, B	Matrices packed by columns in one-dimensional arrays; not destroyed upon return
NA, NB	Two-dimensional vectors giving the number of rows and columns of respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return

Output arguments:

C	Matrix packed by columns in a one-dimensional array. Upon normal return, $C = AB$.
NC	Two-dimensional vector: upon normal return, NC(1) = NA(1) NC(2) = NB(2)

COMMON blocks: None

Error message: If either $NA(2) \neq NB(1)$, $NA(1) < 1$, $NA(1) \times NA(2) < 1$, or $NA(1) \times NB(2) < 1$, the message "DIMENSION ERROR IN MULT NA = _____ _____, NB = _____ _____" is printed, and the program is returned to the calling point.

Field length: 154 octal words (108 decimal)

Subroutine employed by MULT: LNCNT

Subroutines employing MULT: FACTOR, SUM, BILIN, EXPSER, EXPINT, VARANCE, CTROL, TRANSIT, SAMPL, PREFIL, CSTAB, DSTAB, DISCREG, CNTNREG, RICTNWT, ASYMREG, EXPMDFL, IMPMDFL

Comments: None

Subroutine MAXEL

Description: The purpose of MAXEL is to compute the maximum of the absolute values of the elements of a real matrix.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL MAXEL(A,NA,ELMAX)

Input arguments:

A	Matrix packed by columns in a one-dimensional array; not destroyed upon return
NA	Two-dimensional vector giving the number of rows and columns of A: NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return

Output argument:

ELMAX	Scalar. Upon normal return, ELMAX is the maximum of the absolute values of the elements of A.

COMMON blocks: None

Error messages: None

Field length: 27 octal words (23 decimal)

Subroutines employed by MAXEL: None

Subroutines employing MAXEL: SUM, EXPSER, EXPINT, SAMPL, DISCREG, CNTNREG, RICTNWT

Comments: None

Subroutine NORMS

Description: The purpose of NORMS is to compute either the ℓ_1, ℓ_2 (Euclidean), or ℓ_∞ matrix norms for a real $m \times n$ matrix A stored as a variable-dimensioned two-dimensional array. The norms ℓ_1, ℓ_2, and ℓ_∞ are defined, respectively, as

$$\|A\|_1 = \max_{1 \leq k \leq n} \sum_{j=1}^{m} |a_{jk}|$$

$$\|A\|_2 = \left(\sum_{j=1}^{m} \sum_{k=1}^{n} a_{jk}^2 \right)^{1/2}$$

$$\|A\|_\infty = \max_{1 < j < m} \sum_{k=1}^{n} |a_{jk}|$$

Source of software: LaRC Analysis and Computation Division subprogram library

Calling sequence: CALL NORMS(MAXROW,M,N,A,IOPT,RLNORM)

Input arguments:

MAXROW	Maximum first dimension of array A as given in the DIMENSION statement of the calling program
M	Number of rows of matrix A
N	Number of columns of the matrix A
A	Matrix whose norm is desired stored in a two-dimensional array
IOPT	Scalar norm selector: 1 Compute ℓ_1. 2 Compute ℓ_2. 3 Compute ℓ_∞.

Output argument:

RLNORM	Scalar. Upon normal return, RLNORM is the appropriate norm.

COMMON blocks: None

Error messages: None

Field length: 152 octal words (106 decimal)

Subroutines employed by NORMS: None

Subroutines employing NORMS: BILIN, EXPSER, EXPINT, SAMPL, CSTAB

Comments: NORMS can be applied to matrices stored in packed one-dimensional arrays by placing MAXROW = M in the calling sequence.

Subroutine JUXTC

Description: The purpose of JUXTC is to construct a matrix [A,B] from given matrices A and B.

Source of software: VASP (ref. 16) with modifications by Ernest S. Armstrong, LaRC

Calling sequence: `CALL JUXTC(A,NA,B,NB,C,NC)`

Input arguments:

A, B	Matrices packed by columns in one-dimensional arrays; not destroyed upon return
NA, NB	Two-dimensional vectors giving the number of rows and columns of the respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return

Output arguments:

C	Matrix packed by columns in a one-dimensional array. Upon normal return, C = [A,B].
NC	Two-dimensional vector: upon normal return, NC(1) = NA(1) NC(2) = NA(2) + NB(2)

COMMON blocks: None

Error message: If either $NA(1) \neq NB(1)$, $NA(1) < 1$, $NA(1) \times NA(2) < 1$, or $NA(2) + NB(2) < 1$, the message "DIMENSION ERROR IN JUXTC NA = ______ ______, NB = ______ ______" is printed, and the program is returned to the calling point.

Field length: 76 octal words (62 digital)

Subroutine employed by JUXTC: LNCNT

Subroutines employing JUXTC: TESTSTA, CTROL, DSTAB, ASYMREG

Comments: None

Subroutine JUXTR

Description: The purpose of JUXTR is to construct a matrix $\begin{bmatrix} A \\ B \end{bmatrix}$ from given matrices A and B.

Source of software: VASP (ref. 16) with modifications by Ernest S. Armstrong, LaRC

Calling sequence: CALL JUXTR(A,NA,B,NB,C,NC)

Input arguments:

A, B	Matrices packed by columns in one-dimensional arrays; not destroyed upon return
NA, NB	Two-dimensional vectors giving the number of rows and columns of the respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return

Output arguments:

C	Matrix packed by columns in a one-dimensional array. Upon normal return, $C = \begin{bmatrix} A \\ B \end{bmatrix}$.
NC	Two-dimensional vector: upon normal return, NC(1) = NA(1) + NB(1) NC(2) = NA(2)

COMMON blocks: None

Error message: If either $NA(2) \neq NB(2)$, $NA(1) < 1$, $NA(1) \times NA(2) < 1$, or $NA(2) < 1$, the message "DIMENSION ERROR IN JUXTR NA = ______ ______, NB = ______ ______" is printed, and the program is returned to the calling point.

Field length: 144 octal words (100 digital)

Subroutine employed by JUXTR: LNCNT

Subroutines employing JUXTR: BARSTW, CNTNREG

Comments: None

PRIMARY SUBROUTINES FOR ANALYSIS OF CONSTANT LINEAR SYSTEMS

Subroutine FACTOR

Description: The purpose of FACTOR is to compute a real $m \times n$ $(m \leq n)$ matrix D of rank m such that a real $n \times n$ nonnegative definite matrix Q can be factored as

$$Q = D'D$$

The method is first to perform a singular-value (eigenvalue since $Q = Q' \geq 0$) decomposition of Q as

$$Q = PJP'$$

where J is a diagonal matrix of eigenvalues of Q, and then define D as

$$\sqrt{J}\, P'$$

after eliminating those rows of $\sqrt{J}$ with all zero elements. Eigenvalues of Q are considered negligible (zero) if they are less than $\|Q\| \times 10^{-(IAC)}$ using the matrix Euclidean norm.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL FACTOR(Q,NQ,D,ND,IOP,IAC,DUMMY)

Input arguments:

Q	Nonnegative definite matrix packed by columns in a one-dimensional array; not destroyed upon return
NQ	Two-dimensional vector giving the number of rows and columns of Q: NQ(1) = NQ(2) = Number of rows of Q Not destroyed upon return
IOP	Scalar print parameter: 0 Return within printing results. Otherwise Print Q, D, and D'D.
IAC	Scalar parameter to be used for zero test of eigenvalues of Q
DUMMY	Vector of working space for computations with dimension at least $n^2 + n$

Output arguments:

D Matrix packed by columns in a one-dimensional array with dimension at least n^2. Upon normal return, D'D = Q within numerical accuracy.

ND Two-dimensional vector giving number of rows and columns of D: upon normal return,
ND(1) = m
ND(2) = n

COMMON blocks: None

Error messages:

(1) If the singular-value decomposition subroutine SNVDEC fails to converge to a singular value after 30 iterations, the message "IN FACTOR, SNVDEC HAS FAILED TO CONVERGE TO THE ______ SINGULAR VALUE AFTER 30 ITERATIONS" is printed, and the program is returned to the calling point.

(2) If an eigenvalue of Q is greater than but close to the value ZTEST = $\|Q\| \times 10^{-(IAC)}$ (less than 16 × ZTEST), the message "IN FACTOR, THE MATRIX Q SUBMITTED TO SNVDEC IS CLOSE TO A MATRIX OF LOWER RANK USING ZTEST = ______ / IF THE ACCURACY IS REDUCED THE RANK MAY ALSO BE REDUCED / CURRENT RANK = ______" is printed along with the computed singular values, and the computation continues.

Field length: 422 octal words (274 decimal)

Subroutines employed by FACTOR: EQUATE, SNVDEC, LNCNT, TRANP, PRNT, MULT

Subroutines employing FACTOR: None

Comments: None

Subroutine EIGEN

Description: The purpose of EIGEN is to compute all the eigenvalues and selected eigenvectors of a real $n \times n$ matrix A stored as a variable-dimensioned two-dimensional array. The input matrix is first balanced by exact similarity transformations such that the norms of corresponding rows and columns are nearly equal (ref. 17). The balanced matrix is reduced to upper Hessenberg form by stabilized elementary similarity transformations (ref. 18). All of the eigenvalues of the Hessenberg matrix are found by the double shift QR algorithm (ref. 19). The desired eigenvectors of the Hessenberg matrix are then found by the inverse iteration method (ref. 2).

Source of software: LaRC Analysis and Computation Division subprogram library with modifications by Ernest S. Armstrong, LaRC

Calling sequence: `CALL EIGEN(MAX,N,A,ER,EI,ISV,ILV,V,WK,IERR)`

Input arguments:

MAX	Maximum first dimension of the array A as given in the DIMENSION statement of the calling program
N	Actual order of matrix (n)
A	Matrix whose eigenvalues and selected eigenvectors are desired stored in a real two-dimensional array. The contents of this array are destroyed upon return.
ISV	The number of eigenvalues, smallest in absolute value, for which eigenvectors are desired counting complex conjugates
ILV	The number of eigenvalues, largest in absolute value, for which eigenvectors are desired counting complex conjugates
WK	Vector of working space for computations of dimension: 3N if ISV + ILV = 0 N(N+7) otherwise

Output arguments:

ER	One-dimensional real array containing the real parts of the eigenvalues, dimensioned at least N in the calling program
EI	One-dimensional real array containing the imaginary parts of the eigenvalues, dimensioned at least N in the calling program
ISV	On output, ISV is the number of eigenvalues, smallest in absolute value, for which eigenvectors were computed counting complex conjugates

ILV	On output, ILV is the number of eigenvalues, largest in absolute value, for which eigenvectors were computed counting complex conjugates
V	Two-dimensional array containing the eigenvectors, normalized to unit length on a normal return. It suffices to have V dimensioned MAX as first dimension and N as second.
IERR	Integer error code: IERR = 0 Normal return IERR = -J Vectors for the Jth eigenvalue did not converge to an eigenvector. Appropriate column (columns if Jth eigenvalue is complex) of V is set to zero. If failure occurs more than once, the index for the last such occurrence is in IERR. IERR = J The Jth eigenvalue has not been determined after 30 iterations of the QR algorithm.

COMMON blocks: None

Error messages: None. User should test IERR after return.

Field length: 1131 octal words (601 decimal)

Subroutines employed by EIGEN: BALANC, ELMHES, HQR, INVIT, ELMBAK, BALBAK

Subroutines employing EIGEN: BILIN, TESTSTA, CSTAB, DSTAB, CNTNREG, ASYMREG

Comments: Upon normal return, eigenvalues are stored in ascending magnitude with complex conjugates stored with positive imaginary parts first. The eigenvectors are packed and stored in V in the same order as their eigenvalues appear in ER and EI. Only one eigenvector is computed for complex conjugates (for conjugate with positive imaginary part). Upon error exit -J, eigenvalues are correct and eigenvectors are correct for all nonzero vectors. Upon error exit J, eigenvalues are correct but unordered for indices IERR+1,IERR+2,...,N, and no eigenvectors are computed.

EIGEN may be used for matrices packed as a one-dimensional array by setting MAX = N in the calling sequence. If all eigenvectors are desired, set ISV = N and ILV = 0.

Subroutine SYMPDS

Description: The purpose of SYMPDS is to solve the matrix equation,

$$AX = B$$

where A is a symmetric positive definite matrix and B is a matrix of constant vectors. The determinant of A may also be evaluated. Solution is by Cholesky decomposition (ref. 2): A is factored as A = LDL', where L is a unit lower triangular matrix and D is a positive definite diagonal matrix. An option is provided for computing only the Cholesky factorization of A without solving a complete matrix equation. Both A and B are stored as variable-dimensioned two-dimensional arrays.

Source of software: LaRC Analysis and Computation Division subprogram library with modifications by Ernest S. Armstrong, LaRC

Calling sequence: `CALL SYMPDS(MAXN,N,A,NRHS,B,IOPT,IFAC,DETERM,ISCALE,P,IERR)`

Input arguments:

MAXN	The maximum first dimension of A as given in the DIMENSION statement of the calling program
N	The number of rows of A
A	Coefficient matrix stored as a variable-dimensioned two-dimensional array. Two types of input are possible: the first is the undecomposed coefficient matrix, and the second is the Cholesky decomposition A = LDL'. For A in unfactored form, the contents are destroyed upon return. If the factored form of A is input, A contains the upper triangular elements of the coefficient matrix and the elements of the unit lower triangular matrix L except the diagonal elements of L which are understood to be all unity. The reciprocals of the elements of D are input in the array P.
NRHS	The number of column vectors of the matrix B. No data are required if only factorization of A is desired, but NRHS still must appear as an argument of the calling sequence.
B	Two-dimensional array that must have first dimension MAXN and second dimension at least NRHS in the calling program. On input, B contains the elements of the constant vectors, and B is destroyed upon return. If only a factorization of A is required, no data are necessary, but B still must appear as an argument of the calling sequence.
IOPT	Integer for determinant evaluation: 0 Determinant is not evaluated. 1 Determinant is evaluated.

IFAC	Integer specifying whether or not the Cholesky decomposition of the coefficient matrix is to be computed: 0 Cholesky decomposition for the matrix A is to be computed and equation solved. 1 Cholesky decomposition form of A is input so no decomposition is required. 2 Only Cholesky decomposition of A is required.
P	One-dimensional array dimensioned at least N in the calling program. If the unfactored form of A is input, nothing need be input for P. If the factored form of A is input, P contains the reciprocals of the diagonal elements of D.

Output arguments:

A	Upon normal return, the array A contains the original elements of the matrix A and the elements of the unit lower triangular matrix L except the diagonal elements of L which are understood to be all unity.
B	Upon normal return, if a system of equations is to be solved, each solution vector of X is stored over the corresponding constant vector of the input array B.
DETERM, ISCALE	Determinant evaluation parameters: det (A) = DETERM × $10^{(100\times \mathrm{ISCALE})}$
P	Upon normal return, P contains the reciprocals of the diagonal elements of D.
IERR	Error code: 0 Normal return 1 A is not symmetric positive definite.

COMMON blocks: None

Error messages: None. The parameter IERR should be tested after return.

Field length: 373 octal words (251 decimal)

Subroutines employed by SYMPDS: None

Subroutines employing SYMPDS: PREFIL, CNTNREC, RICTNWT, EXPMDFL

Comments: SYMPDS may be used for matrices packed as a one-dimensional array by setting MAXN = N in the calling sequence.

Subroutine GELIM

Description: The purpose of GELIM is to solve the real matrix equation,

$$AX = B$$

where A is required to be square and nonsingular and B is a matrix of constant vectors. Solution is by Gaussian elimination or LU factorization (ref. 2) in which A is factored as

$$PA = LU$$

where L is a unit lower triangular matrix, U is an upper triangular matrix, and P is a permutation matrix representing the row pivotal strategy associated with the LU factorization. Both A and B are stored as variable-dimensioned two-dimensional arrays.

Source of software: LaRC Analysis and Computation Division subprogram library

Calling sequence: CALL GELIM(NMAX,N,A,NRHS,B,IPIVOT,IFAC,WK,IERR)

Input arguments:

NMAX	The maximum first dimension of the array A as given in the DIMENSION statement of the calling program
N	The number of rows of A
A	Coefficient matrix stored as a variable-dimensioned two-dimensional array. Two types of input are possible: the first is the unfactored coefficient matrix, and the second is the triangular factorization A = LU. For A input in factored form, A = (L\U) should be used neglecting the unity elements of L, and the pivotal strategy employed should be input through the array IPIVOT. For A input in unfactored form, input data are destroyed.
NRHS	Number of column vectors of the matrix B
B	Two-dimensional array that must have first dimension NMAX and second dimension at least NRHS in the calling program. On input, B contains the elements of the constant vectors and is destroyed upon return.

IPIVOT An integer array dimensioned at least N in the calling program. If the factored form of A is input, IPIVOT contains the pivotal strategy by the rule,

IPIVOT(I) = J

which states that row J of matrix A was used to pivot for the Ith unknown.

IFAC Factorization parameter:

- 0 Compute L, U, and pivotal strategy.
- 1 Do not compute L, U, and pivotal strategy; the factorization and strategy are input.

WK A one-dimensional array dimensioned at least N in the calling program and used as a work storage array

Output arguments:

A Upon normal return, the unit lower and upper matrices are overstored in A as A = (L\U) neglecting the unity elements L.

B Upon normal return, each solution vector of X is stored over the corresponding constant vector of the input array B.

IPIVOT Upon normal return, IPIVOT contains the pivotal strategy as previously explained.

IERR Singularity test parameter:

- 0 A is nonsingular.
- 1 A is singular.

COMMON blocks: None

Error messages: None. Upon return the parameter IERR should be tested.

Field length: 261 octal words (177 digital)

Subroutine employed by GELIM: DETFAC

Subroutines employing GELIM: BILIN, TRANSIT, DSTAB, CNTNREG

Comments: If an LU factorization of A is desired without a complete equation solution, the subroutine DETFAC may be employed.

GELIM may be used for matrices packed as a one-dimensional array by setting NMAX = N in the calling sequence.

Subroutine SNVDEC

Description: The purpose of SNVDEC is to compute the singular-value decomposition (ref. 2) of a real $m \times n$ $(m \geq n)$ matrix A by performing the factorization,

$$A = UQV'$$

where U is an $m \times n$ matrix whose columns are n orthonormalized eigenvectors associated with the n largest eigenvalues of AA', V is an $n \times n$ matrix whose columns are the orthonormalized eigenvectors associated with the n eigenvalues of A'A, and

$$Q = \text{diag}\ (\sigma_1, \sigma_2, \ldots, \sigma_n)$$

where σ_i $(i = 1,2,\ldots,n)$ are the nonnegative square roots of the eigenvalues of A'A, called the singular values of A. Options are provided for the computation of rank A, singular values of A, an orthonormal basis for the null space of A, the pseudoinverse of A, and the least squares solution to

$$AX = B$$

Both A and B are stored as variable-dimensioned two-dimensional arrays. The computational procedure is described in reference 2 on pages 135-151. Basically, Householder transformations are applied to reduce A to bidiagonal form after which a QR algorithm is used to find the singular values of the reduced matrix. Combining results gives the required construction.

Source of software: LaRC Analysis and Computation Division subprogram library with modifications by Ernest S. Armstrong, LaRC

Calling sequence: CALL SNVDEC(IOP,MD,ND,M,N,A,NOS,B,IAC,ZTEST,Q,V,IRANK, APLUS,IERR)

Input arguments:

IOP Option code:

1 The rank and singular values of A will be returned.
2 The matrices U and V will be returned in addition to the information for IOP = 1.
3 In addition to the information for IOP = 2, the least squares solution to AX = B will be returned.
4 The pseudoinverse of A will be returned in addition to the information for IOP = 2.
5 The least squares solution will be returned in addition to the information for IOP = 4.

MD	The maximum first dimension of the array A as given in the DIMENSION statement of the calling program
ND	Maximum first dimension of the array V
M	The number of rows of A
N	The number of columns of A
A	Matrix stored as a variable-dimensioned two-dimensional array. Input A is destroyed.
NOS	The number of column vectors of the matrix B
B	Two-dimensional array that must have row dimension at least NOS in the calling program. B contains the right sides of the equation to be solved for IOP = 3 or IOP = 5. B need not be input for other options but must appear in the calling sequence.
IAC	The number of decimal digits of accuracy in the elements of the matrix A. This parameter is used in the test to determine zero singular values and thereby the rank of A.

Output arguments:

A	On normal return, A contains the orthogonal matrix U except when IOP = 1.
B	On normal return, B contains the least squares solution for IOP = 3 or IOP = 5.
ZTEST	The zero test computed as $\|A\| \times 10^{-(IAC)}$ using the matrix Euclidean norm except when N = 1. When N = 1, $\text{ZTEST} = 10^{-(IAC)}$
Q	A one-dimensional array of dimension at least N which upon return contains the singular values in descending order
V	A two-dimensional array that must have first dimension ND and second dimension at least N. Upon normal return, this array contains the orthogonal matrix V except when IOP = 1. The last N - IRANK columns of V form a basis for the null space of A.
IRANK	Rank of the matrix A determined as the number of nonzero singular values using ZTEST

APLUS A two-dimensional array of first dimension ND and second dimension at least M. Upon normal return, this array contains the pseudo-inverse of the matrix A. If IOP does not equal 4 or 5, this array need not be dimensioned, but a dummy parameter must appear in the calling sequence.

IERR Error indicator:

IERR = 0	A normal return
IERR = K > 0	The Kth singular value has not been found after 30 iterations of the QR algorithm procedure.
IERR = -1	Using the given IAC, A is close to a matrix which is of lower rank and if the accuracy is reduced, the rank of the matrix may also be reduced.

COMMON blocks: None

Error messages: None. The user should examine IERR after return.

Field length: 2072 octal words (1082 decimal)

Subroutines employed by SNVDEC: None

Subroutines employing SNVDEC: FACTOR, CTROL, CSTAB, DSTAB, DISCREG

Comments: SNVDEC may be applied to matrices stored as one-dimensional arrays by setting MD = M and ND = N in the calling sequence.

The subroutine is internally restricted to $N \le 150$.

Subroutine SUM

Description: The purpose of SUM is to evaluate until convergence the matrix series

$$X = \sum_{i=0}^{\infty} A^i B C^i$$

where A and C are $n \times n$ and $m \times m$ real constant matrices, respectively. The matrix B is real constant and $n \times m$. The series is numerically summed by successively evaluating the partial sum sequence

$$\left.\begin{aligned} S(j+1) &= S(j) + U(j)\ S(j)\ V(j) \\ U(j+1) &= U^2(j) \\ V(j+1) &= V^2(j) \end{aligned}\right\} \qquad (j = 0,1,\ldots)$$

with

$$S(0) = B$$

$$U(0) = A$$

$$V(0) = C$$

The symbol S(j) represents the sum of the first 2^j terms of the power series. Evaluation of the sequence continues until the number of terms evaluated exceeds MAXSUM, which is specified in the COMMON block CONV of subroutine RDTITL, or until convergence is reached. Numerically, convergence of the S(j) sequence is determined by testing the improvement in the element of S(j) of largest magnitude (measured relatively if the magnitude is less than unity, and absolutely otherwise). The sequence is assumed to have converged when this improvement is less than the parameter SUMCV also found in the COMMON block CONV of subroutine RDTITL. A condition sufficient for the convergence of the power series, independent of B, is that each eigenvalue of A and C have complex modulus less than unity. When the series converges, it represents a solution X to

$$X = AXC + B$$

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL SUM(A,NA,B,NB,C,NC,IOP,SYM,DUMMY)

Input arguments:

A, B, C	Compatible matrices packed by columns in one-dimensional arrays. A, B, and C are destroyed upon return.
NA, NB, NC	Two-dimensional vectors giving the number of rows and columns of the respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return
IOP	Print parameter: 0 Do not print results. Otherwise Print input A, B, C, and the sum denoted as X.
SYM	Logical variable: TRUE Indicates A = C' FALSE Otherwise
DUMMY	Vector of working space for computations with dimension at least maximum of (n^2,m^2,2nm)

Output argument:

B	Upon normal return, the sum X is stored in B.

COMMON block: CONV

Error message: If the number of terms in the partial sum sequence S(j) exceeds MAXSUM, the message "IN SUM, THE SEQUENCE OF PARTIAL SUMS HAS EXCEEDED STAGE ______ WITHOUT CONVERGENCE" is printed.

Field length: 367 octal words (247 decimal)

Subroutines employed by SUM: PRNT, MULT, MAXEL, ADD, EQUATE, TRANP, LNCNT

Subroutines employing SUM: BILIN, VARANCE, RICTNWT, EXPMDFL

Comments: A method for scaling the A and C matrices in order to possibly improve convergence is found in reference 15. Also found in reference 15 is a bilinear transformation method for converting the equation,

$$X = AXC + B$$

into one solvable (assuming a unique solution exists) by the subroutine BARSTW of ORACLS.

Subroutine BILIN

Description: The purpose of BILIN is to solve the matrix equation,

$$AX + XB = C$$

where A and B are real constant matrices of dimension $n \times n$ and $m \times m$, respectively. The matrix C is real constant and of dimension $n \times m$. It is assumed that all eigenvalues of A and B have strictly negative real parts. The method of solution employs the bilinear transformation technique described by Smith (ref. 6), wherein it is established that the foregoing X equation is equivalent to

$$X = UXV + W$$

with

$$U = (\beta I_n - A)^{-1}(\beta I_n + A)$$

$$V = (\beta I_m + B)(\beta I_m - B)^{-1}$$

$$W = -2\beta(\beta I_n - A)^{-1}C(\beta I_m - B)^{-1}$$

where β is a scalar greater than zero and I_n and I_m are $n \times n$ and $m \times m$ identity matrices. The eigenvalues of U and V lie within the unit circle in the complex plane; therefore, the series

$$\sum_{i=0}^{\infty} U^i W V^i$$

converges and represents X. The subroutine SUM is used to evaluate the infinite series.

The user has the option of inputing β externally or having the program compute β internally based on the eigenvalues of A and B as follows. Let

$$\lambda_j = a_j + ib_j \qquad (j = 1,2,\ldots,n)$$

be eigenvalues of A ordered so that

$$|\lambda_1| \le |\lambda_2| \le . . . \le |\lambda_n|$$

Choosing β_0 from the equation,

$$\frac{(\beta_0 + a_1)^2 + b_1^2}{(\beta_0 - a_1)^2 + b_1^2} = \frac{(\beta_0 + a_n)^2 + b_n^2}{(\beta_0 - a_n)^2 + b_n^2}$$

gives

$$\beta_0^2 = \frac{a_1\left(a_n^2 + b_n^2\right) - a_n\left(a_1^2 + b_1^2\right)}{(a_n - a_1)}$$

When $a_n - a_1 = 0$ or $\beta_0^2 \le 0$, instead set

$$\beta_0 = \frac{\sum_{j=1}^{n} \left(a_j^2 + b_j^2\right)^{1/2}}{n}$$

If the eigenvalue computation fails, set

$$\beta_0 = 2\|A\|$$

using the ℓ_1 matrix norm (see subroutine NORMS). Similarly, compute β_1 based on B and put

$$\beta = \frac{1}{2}(\beta_0 + \beta_1)$$

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL BILIN(A,NA,B,NB,C,NC,IOP,BETA,SYM,DUMMY)

Input arguments:

A, B, C	Compatible matrices packed by columns in one-dimensional arrays. C is destroyed upon return, but A and B are not. If A = B', no data need be entered for B and NB, but their symbols must still appear in the argument list.
NA, NB, NC	Two-dimensional vectors giving the number of rows and columns of the respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return
IOP	Two-dimensional parameter vector: IOP(1) = 0 Do not print results. Otherwise Print A, B, C, BETA, and X. IOP(2) = 0 Use input value BETA for β. Otherwise Compute β as previously described.
BETA	Scalar value β for numerical conditioning. No input is required if IOP(2) is nonzero.
SYM	Logical variable: TRUE Indicates A = B' FALSE Otherwise
DUMMY	Vector of working space for computations with dimension at least $(4p^2 + 2p)$ with $p = \max(n,m)$

Output argument:

C	Upon normal return, the solution X is stored in C.

COMMON blocks: None

Error messages:

(1) If β is computed internally and the eigenvalue computation for A fails, the message "IN BILIN, THE ______ EIGENVALUE OF A HAS NOT BEEN DETERMINED AFTER 30 ITERATIONS" is printed. A similar message is printed if the eigenvalue computation for B fails.

(2) If the $(\beta I_n - A)^{-1}$ computation fails because of singularity, the message "IN BILIN, THE MATRIX (BETA)I - A IS SINGULAR, INCREASE BETA" is printed. A similar message is printed if the $(\beta I_m - B)^{-1}$ computation fails.

Field length: 1512 octal words (842 decimal)

Subroutines employed by BILIN: LNCNT, PRNT, TRANP, EQUATE, EIGEN, SCALE, MULT, SUM, GELIM, NORMS

Subroutines employing BILIN: VARANCE, CSTAB, RICTNWT

Comments: For A and B not satisfying the requirement of eigenvalues with strictly negative real parts, but still admitting a unique solution to

$$AX + XB = C$$

the subroutine BARSTW may be applied.

Subroutine BARSTW

Description: The purpose of BARSTW is to solve the matrix equation

$$AX + XB = C$$

where A and B are real constant matrices of dimension $n \times n$ and $m \times m$, respectively. The matrix C is real constant and of dimension $n \times m$. It is assumed that

$$\lambda_i^A + \lambda_j^B \neq 0 \qquad (i = 1,2,\ldots,n;\quad j = 1,2,\ldots,m)$$

where λ_i^A and λ_j^B are eigenvalues of A and B, respectively. The matrix equation then has a unique solution X. The method of solution is based on transforming A and B to real Schur form as described by Bartels and Stewart (ref. 5).

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: `CALL BARSTW(A,NA,B,NB,C,NC,IOP,SYM,EPSA,EPSB,DUMMY)`

Input arguments:

A, B, C	Compatible matrices packed by columns in one-dimensional arrays. C is destroyed upon return, but A and B are not. If A = B' and C = C', no data need be entered for B and NB, but their symbols should appear in the argument list.
NA, NB, NC	Two-dimensional vectors giving the number of rows and columns of the respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return
IOP	Print option parameter: 0 Do not print. Otherwise Print A, B, C, and X.
SYM	Logical variable: TRUE Indicates A = B' and C = C' FALSE Otherwise
EPSA, EPSB	Convergence criteria for the reduction of A and B to Schur form. EPSA should be set slightly smaller than $10^{(-N)}$, where N is the number of significant digits in the elements of A. EPSB is similarly defined from B.

DUMMY	Vector of working space for computations with dimensions at least $2(n + 1)^2$ for SYM = TRUE $2(n + 1)^2 + 2(m + 1)^2$ for SYM = FALSE

Output argument:

C	Upon normal return, the solution X is stored in C.

COMMON blocks: None

Error message: If the reduction to Schur form fails, the message "IN BARSTW, EITHER THE SUBROUTINE AXPXB OR ATXPXA WAS UNABLE TO REDUCE A OR B TO SCHUR FORM" is printed, and the program is returned to the calling point.

Field length: 521 octal words (337 decimal)

Subroutines employed by BARSTW: PRNT, TRANP, EQUATE, NULL, JUXTR, AXPXB, ATXPXA, LNCNT

Subroutines employing BARSTW: VARANCE, CSTAB, DSTAB, RICTNWT, EXPMDFL

Comments: If a set of equations is to be solved for a collection of C matrices with the same A and B or the Schur forms are desired along with X, the subroutines AXPXB and ATXPXA should be applied directly.

Subroutine TESTSTA

Description: The purpose of TESTSTA is to compute and test the eigenvalues of a constant real matrix A for stability relative to a parameter α in either the continuous or digital sense. In the continuous case, the matrix A is classified as stable if the real part of each eigenvalue is strictly less than α. Otherwise, A is classified as unstable relative to α. In the discrete case, the matrix A is classified as stable if the complex modulus of each eigenvalue is less than α. Otherwise, A is declared unstable relative to α.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL TESTSTA(A,NA,ALPHA,DISC,STABLE,IOP,DUMMY)

Input arguments:

A	Square real matrix packed by columns in a one-dimensional array; not destroyed upon return
NA	Two-dimensional vector giving the number of rows and columns of A: NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return
ALPHA	Scalar α used for testing relative stability
DISC	Logical variable: TRUE Test for stability in the discrete sense. FALSE Test for stability in the continuous sense.
IOP	Print parameter: 0 Do not print results. Otherwise Print A, the eigenvalues of A, and stability results.
DUMMY	Vector of working space for computations with dimension at least $(n^2 + 5n)$ where n is the order of A

Output arguments:

STABLE	Logical variable: upon normal return, STABLE = TRUE if the stability tests are satisfied; otherwise, STABLE = FALSE.
DUMMY	Upon a normal return, the eigenvalues of A are stored in the first 2n elements of DUMMY and packed by columns as a n × 2 matrix. For DISC = TRUE, the moduli of the eigenvalues of A appear in the next n elements.

COMMON blocks: None

Error message: If the computation of the eigenvalues of A fails, the message "IN TESTSTA, THE ______ EIGENVALUE OF A HAS NOT BEEN FOUND AFTER 30 ITERATIONS" is printed, and the program is returned to the calling point.

Field length: 366 octal words (246 decimal)

Subroutines employed by TESTSTA: EQUATE, EIGEN, JUXTC, PRNT

Subroutine employing TESTSTA: ASYMREG

Comments: None

Subroutine EXPSER

Description: The purpose of EXPSER is to evaluate the matrix exponential

$$e^{AT}$$

for a real square matrix A and scalar T. Computation is based on the finite-series algorithm described by Källström (ref. 20). The matrix AT is scaled by $1/2^k$ where k is a positive integer chosen so that the scaled matrix has eigenvalues within the unit circle in the complex plane. The series algorithm is applied to the scaled matrix until the series converges. Numerically, convergence is assumed to have occurred when the improvement in the element of the finite series of largest magnitude (measured relatively if the magnitude is less than unity, and absolutely otherwise) is less than the parameter SERCV found in the COMMON block CONV of subroutine RDTITL. Finally, the desired matrix exponential is reconstructed from the exponential of the scaled matrix.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL EXPSER(A,NA,EXPA,NEXPA,T,IOP,DUMMY)

Input arguments:

A	Square matrix packed by columns in a one-dimensional array; not destroyed upon return
NA	Two-dimensional vector giving the number of rows and columns of A: NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return
T	Scalar parameter
IOP	Print parameter: 0 Do not print results. Otherwise Print A, T, and e^{AT}.
DUMMY	Vector of working space for computations with dimension at least $2n^2$ where n is the order of A

Output arguments:

EXPA	Upon a normal return, a square matrix packed by columns in a one-dimensional array containing the matrix exponential e^{AT}
NEXPA	Two-dimensional column vector giving the number of rows and columns of e^{AT}: NEXPA(1) = NA(1) NEXPA(2) = NA(2)

COMMON block: CONV

Error messages:

(1) The integer k is tested, and if found to be negative, the message "ERROR IN EXPSER, K IS NEGATIVE" is printed, and the program returned to the calling point.

(2) If k increases to 1000, the message "ERROR IN EXPINT, K = 1000" is printed, and the program is returned to the calling point.

Field length: 673 octal words (443 decimal)

Subroutines employed by EXPSER: MAXEL, UNITY, TRCE, EQUATE, NORMS, SCALE, ADD, MULT, PRNT

Subroutines employing EXPSER: TRANSIT, SAMPL, CNTNREG

Comments: None

Subroutine EXPADE

Description: The purpose of EXPADE is to compute the matrix exponential e^A, where A is a real square matrix stored as a variable-dimensioned two-dimensional array. Computation is by the method of Padé approximation (ref. 21). The matrix is first scaled by a power of 2 chosen so that the eigenvalues of the scaled matrix are within the unit circle in the complex plane. The exponential is computed for this scaled matrix using the approximation given by the ninth diagonal term in the Padé table for exponential approximations. The exponential for the original matrix is then reconstructed from the exponential of the scaled matrix.

Source of software: LaRC Analysis and Computation Division subprogram library with modifications by Ernest S. Armstrong, LaRC

Calling sequence: `CALL EXPADE(MAX,N,A,EA,IDIG,WK,IERR)`

Input arguments:

MAX	Maximum first dimension of A as given in the DIMENSION statement of the calling program
N	Order of matrix A
A	Matrix stored in a two-dimensional array with first dimension MAX and second at least N; not destroyed upon return
IDIG	An estimate of the number of accurate digits in the largest elements of absolute value of e^A
WK	Vector of working space for computations, dimensioned at least $n^2 + 8n$ where n is the order of A

Output arguments:

EA	Matrix stored in a two-dimensional array with first dimension MAX and second at least N. Upon normal return, EA contains the matrix exponential e^A.
IERR	Error parameter: 0 Normal return 1 The sum of the absolute values of the elements of A is too large for any accuracy. 2 The Padé denominator matrix is singular. Singularly should not occur with exact arithmetic.

COMMON blocks: None

Error messages: None. The user should examine the parameter IERR upon return.

Field length: 525 octal words (341 decimal)

Subroutine employed by EXPADE: GAUSEL

Subroutines employing EXPADE: None

Comments: None

Subroutine EXPINT

Description: The purpose of EXPINT is to compute both the matrix exponential

$$e^{AT}$$

and the integral

$$\int_0^T e^{As}\,ds$$

for a square real matrix A and scalar T. Computation is based on the finite-series algorithm described by Källström (ref. 20). The matrix AT is scaled by $1/2^k$ where k is a positive integer chosen so that the scaled matrix has eigenvalues within the unit circle in the complex plane. The series algorithms are applied to the scaled matrix until convergence occurs. Numerically, convergence is assumed to have occurred in each series when the improvement in the element of the finite series of largest magnitude (measured relatively if the magnitude is less than unity, and absolutely otherwise) is less than the parameter SERCV found in the COMMON block CONV of subroutine RDTITL. Finally, the desired matrix exponential and the integral are reconstructed from the results using the scaled matrix.

Source of software: Ernest S. Armstrong, LaRC; based on a similar routine in VASP (ref. 16)

Calling sequence: CALL EXPINT(A,NA,B,NB,C,NC,T,IOP,DUMMY)

Input arguments:

A	Square matrix packed by columns in a one-dimensional array; not destroyed upon return
NA	Two-dimensional vector giving the number of rows and columns of A: NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return
T	Scalar parameter
IOP	Print parameter: 0 Do not print results. Otherwise Print A, T, e^{AT}, and $\int_0^T e^{As}\,ds$
DUMMY	Vector of working space for computations, dimensioned at least $2n^2$ where n is the order of A

Output arguments:

B Upon normal return, square matrix packed by columns in a one-dimensional array containing the matrix exponential

$$e^{AT}$$

NB Two-dimensional column vector containing the row and column size of e^{AT}:
NB(1) = NA(1)
NB(2) = NA(2)

C Upon normal return, square matrix packed by columns in a one-dimensional array containing the matrix

$$\int_0^T e^{As}\,ds$$

NC Two-dimensional column vector giving the number of rows and columns of

$$\int_0^T e^{As}\,ds$$

That is,
NC(1) = NA(1)
NC(2) = NA(2)

COMMON block: CONV

Error messages:

(1) If k is found to be negative, the message "ERROR IN EXPINT, K IS NEGATIVE" is printed, and the program is returned to the calling point.

(2) If k increases to 1000, the message "ERROR IN EXPINT, K = 1000" is printed, and the program is returned to the calling point.

Field length: 744 octal words (484 decimal)

Subroutines employed by EXPINT: NORMS, SCALE, UNITY, ADD, EQUATE, MULT, LNCNT, PRNT, MAXEL

Subroutines employing EXPINT: TRANSIT, SAMPL

Comments: None

Subroutine VARANCE

Description: The purpose of VARANCE is to compute the steady-state variance matrix of the state of the continuous or discrete linear time-invariant system:

Continuous

$$\dot{x}(t) = A\,x(t) + G\,\eta(t)$$

Discrete

$$x(i + 1) = A\,x(i) + G\,\eta(i)$$

where A is an $n \times n$ asymptotically stable matrix, G is an $n \times m$ $(m \leq n)$ matrix, and η is a zero-mean white-noise process with continuous intensity or discrete variance Q. Following reference 4, the intensity of a continuous white-noise process is defined as the coefficient matrix of the δ-function in the covariance formula. The steady-state variance matrices, each denoted by W, satisfy the Liapunov equations:

Continuous

$$AW + WA' = -GQG'$$

Discrete

$$W = AWA' + GQG'$$

The program provides the option of solving the continuous Liapunov equation by either subroutine BILIN or subroutine BARSTW. The discrete Liapunov equation is solved by using subroutine SUM. The computational parameter EPSA used by BARSTW is specified internally as EPSAM found in the COMMON block TOL of subroutine RDTITL.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL VARANCE(A,NA,G,NG,Q,NQ,W,NW,IDENT,DISC,IOP,DUMMY)

Input arguments:

A, G, Q	Matrices packed by columns in one-dimensional arrays; not destroyed upon normal return except for Q which is replaced by GQG'. Storage for Q should be prescribed accordingly in the calling program.

NA, NG, NQ Two-dimensional vectors giving number of rows and columns of the respective matrices; for example,
NA(1) = Number of rows of A
NA(2) = Number of columns of A
Not destroyed upon return

IDENT Logical variable:

TRUE	If G is an identity matrix
FALSE	Otherwise

If IDENT = TRUE, no data are required in G and NG, but the variables must still appear as arguments of the calling sequence.

DISC Logical variable:

TRUE	If the discrete case is solved
FALSE	For the continuous case

IOP Three-dimensional option vector:

IOP(1) = 0	Do not print results.
Otherwise	Print A, G, GQG', Q, and W.
IOP(2) = 0	Do not print from Liapunov equation subroutines employed.
Otherwise	Print from these subroutines.
IOP(3) = 0 and DISC = FALSE	Solve the Liapunov equation by subroutine BARSTW.
IOP(3) ≠ 0 and DISC = FALSE	Solve the Liapunov equation by subroutine BILIN.

IOP(3) is not required if DISC = TRUE.

DUMMY Vector of working spaces for computations dimensioned at least:

$2(n + 1)^2$	for DISC = FALSE and IOP(3) = 0
$4n^2 + 2n$	for DISC = FALSE and IOP(3) ≠ 0
$4n^2$	for DISC = TRUE

Output arguments:

W Upon normal return, matrix packed by columns in a one-dimensional array holding the steady-state variance matrix

NW Two-dimensional vector giving number of rows and columns of W: upon normal return,
NW(1) = NA(1)
NW(2) = NA(2)

COMMON block: TOL

Error messages: None

Field length: 513 octal words (331 decimal)

Subroutines employed by VARANCE: PRNT, LNCNT, MULT, TRANP, BILIN, BARSTW, SUM

Subroutines employing VARANCE: None

Comments: In determining the storage allocation requirements for VARANCE, it was assumed that $m \leq n$. For $m > n$, the matrix GQG' should be computed externally and input as Q with IDENT = TRUE.

Subroutine CTROL

Description: The purpose of CTROL is to evaluate the controllability matrix

$$C = \left[B, AB, \ldots, A^{n-1}B\right]$$

for a real constant (A,B) pair. The matrix A is $n \times n$ and B is $n \times r$ with $r \leq n$. Options are provided to compute both the rank and singular values of C along with the controllability canonical form (ref. 4) for the (A,B) pair. For the optional computations, the singular-value decomposition algorithm found in subroutine SNVDEC is applied to factor

$$\tilde{C}' = \left[B, AB, \ldots, A^{n-r}B\right]$$

as

$$\tilde{C}' = UQV'$$

or

$$\tilde{C} = VQU'$$

The number of nonzero elements on the diagonal of the $n \times n$ matrix Q determines the rank of C. Assuming C has rank $\ell \leq n$, the first ℓ columns of V form an orthonormal basis for the range space of C and the next $(\ell + 1)$ to n columns form an orthonormal basis for the orthogonal complement to the range space of C. Hence, the pair $(V'AV, V'B)$ represents the controllability canonical form for the original (A,B) pair.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL CTROL(A,NA,B,NB,C,NC,IOP,IAC,IRANK,DUMMY)

Input arguments:

A, B	Matrices packed by columns in one-dimensional arrays; not destroyed upon return
NA, NB	Two-dimensional vectors giving the number of rows and columns of respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return

IOP Five-dimensional option vector:

IOP(1) = 0	Do not print A, B, and C.
Otherwise	Print A, B, and C.
IOP(2) = 0	Return after computing C.
Otherwise	Compute rank of C.
IOP(3) = 0	Do not print rank of C, zero test employed to determine rank, and singular values of C.
Otherwise	Print these data.
IOP(4) = 0	Return after rank computation.
Otherwise	Compute the controllability canonical form.
IOP(5) = 0	Return without printing controllability form.
Otherwise	Print V'AV, V'B, and V' before returning.

IAC Parameter used to specify zero test for rank computation. Singular values are considered zero if they do not exceed

ZTEST = $\|\tilde{C}\| \times 10^{-(IAC)}$ using the matrix Euclidean norm.

DUMMY Vector of working space for computations dimensioned at least:

$nr(n+1) = K$	To compute C only
$\max\left[K, n^2+n+nr(n-r+1)\right] = L$	To compute rank of C
$\max(L, 3n^2)$	To compute controllability canonical form

Output arguments:

C Matrix packed by columns in a one-dimensional array dimensioned at least n^2r. Upon normal return, C contains the controllability matrix for the (A,B) pair.

NC Two-dimensional vector giving the number of rows and columns of C: upon normal return,

NC(1) = NA(1) = n

NC(2) = NA(1) × NB(1) = nr

IRANK Upon normal return, scalar giving the rank of C

DUMMY Upon normal return for rank computation only, the first $nr(n-r+1)$ elements contain the matrix U, the next n elements contain the singular values of $\tilde{C}'$, and the next n^2 elements contain the matrix V. Upon normal return for the canonical form computation, the first n^2 elements contain the matrix V', the next nr elements contain the matrix V'B, and, after the first $2n^2$ elements of DUMMY, the next n^2 elements contain the matrix V'AV. All matrices are packed by columns into the corresponding sections of the one-dimensional array DUMMY.

COMMON blocks: None

Error messages:

(1) If SNVDEC fails to compute the singular values of $\tilde{C}$, the message "IN CTROL, SNDVEC HAS FAILED TO CONVERGE TO THE _____ SINGULAR VALUE AFTER 30 ITERATIONS" is printed, and the program is returned to the calling point.

(2) If an eigenvalue of Q is greater than but close to ZTEST (see subroutine SNVDEC), the message "IN CTROL, THE MATRIX SUBMITTED TO SNVDEC USING ZTEST = _____ IS CLOSE TO A MATRIX WHICH IS OF LOWER RANK / IF THE ACCURACY IS REDUCED THE RANK MAY ALSO BE REDUCED/CURRENT RANK = _____" is printed, and the computation continues.

Field length: 656 octal words (430 decimal)

Subroutines employed by CTROL: EQUATE, MULT, JUXTC, PRNT, LNCNT, TRANP, SNVDEC

Subroutines employing CTROL: None

Comments: Employing duality theory, CTROL may also be used to compute the reconstructibility canonical form (ref. 4). By combining controllability and reconstructibility results from CTROL, the full canonical structure theorem (ref. 22) can be implemented.

Subroutine TRANSIT

Description: The purpose of TRANSIT is to compute and print the transient response of the time-invariant continuous or discrete system.

Continuous

$$\dot{X}(t) = AX(t) + B\,U(t)$$

Discrete

$$X(i + 1) = AX(i) + B\,U(i)$$

together with Y and U where

$$Y = HX + GU$$

$$U = -FX + V$$

for given

$$X(0) = X_0$$

from the initial time or stage zero to an input final time or stage. The matrices A, B, X, and H are dimensioned $n \times n$, $n \times r$ $(r \leq n)$, $n \times p$ $(p \leq n)$, and $m \times n$ $(m \leq n)$, respectively, with the other matrices compatible. If the matrix $(A - BF)$ is asymptotically stable in the appropriate continuous or discrete sense, the steady-state value of X given by

Continuous

$$X = -(A - BF)^{-1}BV$$

Discrete

$$X = \left[I - (A - BF)\right]^{-1}BV$$

where I is an identity matrix, is computed and printed except when V is a null matrix.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL TRANSIT(A,NA,B,NB,H,NH,G,NG,F,NF,V,NV,T,X,NX,DISC,STABLE,IOP,DUMMY)

Input arguments:

A, B, H, G, F, V	Compatible matrices packed by columns in one-dimensional arrays; not destroyed upon normal return
NA, NB, NH, NG, NF, NV	Two-dimensional vectors giving the number of rows and columns of respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return
T	Two-dimensional vector: T(1) is the final time in the continuous case given as an integer multiple of T(2). T(2) is a time point within $[0,T(1)]$; printing is at multiples of T(2). The input T is not required if the response to a discrete system is to be computed, but the argument must still appear in the calling sequence; not destroyed upon normal return.
X	Matrix packed by columns in a one-dimensional array containing the initial value X_0 at time or stage zero. X is destroyed upon normal return.
NX	Two-dimensional vector giving the number of rows and columns of X: NX(1) = NA(1) NX(2) = NU(2) Not destroyed upon normal return
DISC	Logical variable: TRUE If the response to the discrete system is required FALSE For the response to the continuous system
STABLE	Logical variable: TRUE If (A - BF) is asymptotically stable in the appropriate sense STABLE = FALSE Otherwise

IOP Four-dimensional option vector:

IOP(1) = 0	If H is a null matrix
IOP(1) = 1	If H is an identity matrix
IOP(1) $\neq$ 0 or 1	For other H matrices
IOP(2) = 0	If G is a null matrix
IOP(2) = 1	If G is an identity matrix
IOP(2) $\neq$ 0 or 1	For other G matrices
IOP(3) = 0	If V is a null matrix
IOP(3) = 1	If V is an identity matrix
IOP(3) $\neq$ 0 or 1	For other V matrices

IOP(4) is the terminal stage for the response to a discrete system; not required if the continuous system response is computed (DISC = FALSE).

DUMMY Vector of working space for computations, dimensioned at least $7n^2$ where n is the order of A.

Output arguments:

X Upon normal return, the value of X at time T(1) or state IOP(4)

DUMMY Upon normal return, the first np elements of DUMMY contain the steady-state value of X, packed by columns in a one-dimensional array, when STABLE = TRUE and IOP(3) $\neq$ 0.

COMMON blocks: None

Error message: When computation of the continuous steady-state X values is attempted and (A - BF) is found to be singular, the message "IN TRANSIT, THE MATRIX A-BF SUBMITTED TO GELIM IS SINGULAR" is printed, and the program is returned to the calling point. For the discrete case, a similar message concerning the matrix [I - (A - BF)] is also printed.

Field length: 1530 octal words (856 decimal)

Subroutines employed by TRANSIT: PRNT, EXPSER, EXPINT, MULT, EQUATE, LNCNT, SCALE, ADD, TRANP, GELIM, UNITY, SUBT

Subroutines employing TRANSIT: None

Comments: When the matrices H, G, or V take on their special null or identity matrix values, no data need be input, but they must still appear as arguments of the calling sequence.

PRIMARY SUBROUTINES FOR IMPLEMENTING LQG CONTROL LAW DESIGN

Subroutine SAMPL

Description: The purpose of SAMPL is to compute the matrix functions,

$$\tilde{Q}(T) = \int_0^T e^{A'\tau} Q e^{A\tau} \, d\tau$$

$$W(T) = 2 \int_0^T e^{A'} Q \; H(\tau,0) \, d\tau$$

$$\tilde{R}(T) = \int_0^T \left[R + H'(\tau,0) \; Q \; H(\tau,0)\right] d\tau$$

where

$$H(t,0) = \int_0^t e^{A\tau} B \, d\tau$$

for constant real matrices A, B, $Q = Q' \geq 0$, $R' = R > 0$, and scalar $T > 0$. The dimensions of A and B are $n \times n$ and $n \times r$ $(r \leq n)$, respectively. Other matrices have compatible dimensions. The program has the option of computing $\tilde{Q}(T)$ without evaluating $W(T)$ and $\tilde{R}(T)$.

The matrix $\tilde{Q}$ is the reconstructibility Gramian (ref. 4) for the system

$$\left.\begin{aligned} \dot{x}(t) &= A \, x(t) \\ y(t) &= D \, x(t) \\ Q &= D'D \end{aligned}\right\} \qquad (0 \leq t \leq T)$$

The set of matrices $\tilde{Q}$, W, and $\tilde{R}$ occur naturally in the optimal sampled-data linear regulator problem (refs. 23 and 24). Given the time-invariant linear system,

$$\dot{x}(t) = A \, x(t) + B \, u(t)$$

the optimal sampled-data regulator problem (OSR) occurs when $u(t)$ $(0 \le t \le t_f < \infty)$ is required to minimize

$$J = x'(t_f)\, S\, x(t_f) + \int_0^{t_f} \left[x'(\tau)\, Q\, x(\tau) + u'(\tau)\, R\, u(\tau)\right] d\tau$$

subject to the restrictions that $u(t)$ be constant over subintervals $[t_i, t_{i+1}]$ $(i = 0,1,\ldots,N-1)$, $0 < t_0 < t_1 < t_2 \ldots < t_N = t_f$ within the interval $[0, t_f]$, and $S = S' \ge 0$. The OSR problem transforms directly into an optimal discrete regulator problem in which $u(t_i)$ $(i = 0,1,\ldots,N-1)$ is chosen to minimize

$$J = x'(t_f)\, S\, x(t_f) + \sum_{i=0}^{N-1} \Big[x'(t_i)\, \tilde{Q}(\Delta t_i)\, x(t_i)$$

$$+ x'(t_i)\, W(\Delta t_i)\, u(t_i) + u'(t_i)\, \tilde{R}(\Delta t_i)\, u(t_i)\Big]$$

with

$$\Delta t_i = t_{i+1} - t_i$$

and

$$x(t_{i+1}) = e^{A\Delta t_i}\, x(t_i) + H(\Delta t_i, 0)\, u(t_i)$$

The subroutine SAMPL computes the weighting matrices for the OSR problem for the case $\Delta t_i = T$.

Computation is based on the finite-series algorithm described by Armstrong and Caglayan (ref. 13). The matrix AT is scaled by $1/2^k$ where k is a positive integer chosen so that the scaled matrix has eigenvalues within the unit circle in the complex plane. The algorithm (ref. 13) is applied to the scaled matrix until convergence occurs. Numerically, convergence is assumed to have occurred in each series when the improvement in the element of largest magnitude (measured relatively if the magnitude is less than unity, and absolutely otherwise) is less than the parameter SERCV found in the COMMON block CONV of subroutine RDTITL. Afterward, the desired OSR weighting matrices are reconstructed from the results with the scaled AT matrix, as described in reference 13.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL SAMPL(A,NA,B,NB,Q,NQ,R,NR,W,NW,T,IOP,DUMMY)

Input arguments:

A, B, Q, R	Matrices packed by columns in one-dimensional arrays. A and B are not destroyed upon return, but Q and R are. If only $\tilde{Q}$ is to be computed, data for B and R need not be input, but the related arguments should still appear in the calling sequence.
NA, NB, NQ, NR	Two-dimensional vectors giving the number of rows and columns of respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon normal return
T	Positive scalar parameter
IOP	Two-dimensional option vector: IOP(1) = 0 Do not print results of computation. Otherwise Print input and computed results. IOP(2) = 0 Solve for the reconstructibility Gramian only. Otherwise Solve for the complete set of OSR weighting matrices.
DUMMY	Vector of working space for computations with dimension at least: $4n^2$ for IOP(2) = 0 $7n^2$ for IOP(2) ≠ 0

Output arguments:

Q	Upon normal return, the matrix $\tilde{Q}(T)$ is stored in Q and packed by columns in the one-dimensional array.
W	Matrix packed by columns in a one-dimensional array with dimension at least nr. Upon normal return, W contains the matrix W(T) if computation is required. If the computation of W(T) is not required, W is not needed, but the argument should still appear in the calling sequence.
NW	Two-dimensional vector giving, upon normal return, the number of rows and columns of W(T): NW(1) = NA(1) NW(2) = NB(2)
R	Upon normal return, the matrix $\tilde{R}(T)$ is stored in R and packed by columns in the one-dimensional array.

COMMON block: CONV

Error message: If k reaches 1000 in the scaling of AT, the message "ERROR IN SAMPL, K = 1000" is printed, and the program is returned to the calling point.

Field length: 2521 octal words (1361 decimal)

Subroutines employed by SAMPL: PRNT, LNCNT, NORMS, SCALE, EQUATE, MULT, TRANP, ADD, MAXEL, EXPSER, EXPINT

Subroutines employing SAMPL: None

Comments: The variance matrix for the state of the time-invariant linear system

$$\dot{x}(t) = A\,x(t) + B\,\eta(t)$$

driven by white noise η with intensity V is given at time T by

$$G(T) = e^{AT}\,G(0)\,e^{A'T} + \int_0^T e^{A\tau}BVB'e^{A'\tau}\,d\tau$$

and can be computed through the matrix exponential and SAMPL subroutines of ORACLS. For the second term in $G(T)$, use SAMPL and compute the reconstructibility Gramian with A replaced by A' and $Q = BVB'$.

Subroutine PREFIL

Description: The purpose of PREFIL is to compute an $r \times n$ $(r \leq n)$ matrix F which, when used in the vector equation,

$$u = -Fx + v$$

eliminates the cross-product term in the quadratic scalar function,

$$x'Qx + x'Wu + u'Ru$$

where $Q = Q' \geq 0$, W, and $R = R' > 0$ are constant matrices. Specifically,

$$F = R^{-1} \frac{W}{2}$$

and, after substitution, the quadratic function becomes

$$x'\tilde{Q}x + v'Rv$$

where

$$\tilde{Q} = Q - \frac{W}{2} F$$

If the transformation is also applied to a continuous or discrete linear time-invariant system,

Continuous

$$\dot{x} = Ax + Bu$$

Discrete

$$x(i+1) = A\,x(i) + B\,u(i)$$

the closed-loop response matrix is

$$\tilde{A} = A - BF$$

Options are provided within PREFIL to compute both $\tilde{Q}$ and $\tilde{A}$ in addition to F.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL PREFIL(A,NA,B,NB,Q,NQ,W,NW,R,NR,F,NF,IOP,DUMMY)

Input arguments:

A, B, Q, W, R	Matrices packed by columns in one-dimensional arrays. Inputs B and R are not destroyed upon normal return. If only the matrix F is required, input A, B, and Q are not required but must still appear in the calling sequence. Similarly, if only F and $\tilde{Q}$ are required, input B is not required, but B must still appear in the calling sequence.
NA, NB, NQ, NW, NR	Two-dimensional vectors giving the number of rows and columns in the respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon return
IOP	Three-dimensional option vector: IOP(1) = 0 Do not print results. Otherwise Print input data, F, and $(\tilde{Q}, \tilde{A})$ when computed. IOP(2) = 0 Do not compute $\tilde{Q}$. Otherwise Compute F and $\tilde{Q}$ and return. IOP(3) = 0 Do not compute $\tilde{A}$. Otherwise Compute F and $\tilde{A}$ and return.
DUMMY	Vector of working space for computations, dimensioned at least $n^2 + r$ where n and r are the order of A and R, respectively

Output arguments:

A	If $\tilde{A}$ is computed, the A array contains, upon normal return, the matrix $\tilde{A}$ packed by columns.
Q	If $\tilde{Q}$ is computed, the Q array contains, upon normal return, the matrix $\tilde{Q}$ packed by columns.
F	Upon normal return, the matrix F packed by columns in a one-dimensional array of dimension at least nr
NF	Upon normal return, a two-dimensional vector giving the number of rows and columns of F: NF(1) = NR(1) NF(2) = NA(1)

COMMON blocks: None

Error message: If the matrix R is found not to be symmetric positive definite by the subroutine SYMPDS, the message "IN PREFIL, THE MATRIX R IS NOT SYMMETRIC POSITIVE DEFINITE" is printed, and the program is returned to the calling point.

Field length: 457 octal words (303 decimal)

Subroutines employed by PREFIL: PRNT, LNCNT, TRANP, SCALE, EQUATE, SYMPDS, MULT, SUBT

Subroutine employing PREFIL: IMPMDFL

Comments: Subroutines which follow provide solution algorithms for LQG problems having a quadratic performance index without cross-product terms. Problems with such terms may be transformed into equivalent problems without cross products by performing a control variable transformation (ref. 4):

$$u = -Fx + v$$

where u is the original control, x is the state vector, and v is the new control with F computed from PREFIL.

Subroutine CSTAB

Description: The purpose of CSTAB is to compute a gain matrix F which, when used in the control law

$$u = -Fx$$

and applied to the stabilizable linear time-invariant continuous system,

$$\dot{x} = Ax + Bu$$

produces a closed-loop response matrix (A - BF) whose eigenvalues lie in the complex left half-plane. The primary use for CSTAB in ORACLS is to generate a stabilizing gain matrix for initializing the quasilinearization method for solving the continuous steady-state Riccati equation (ref. 9). The matrix (A - BF) computed here has some interesting root-locus properties (ref. 25) which may make the control law applicable in other areas.

Computation follows the method described by Armstrong (ref. 11). The matrices A and B are of dimension n × n and n × r $(r \leq n)$, respectively. A scalar parameter $\beta > 0$ is first selected so that the matrix

$$\tilde{A} = -(A + \beta I)$$

has eigenvalues in the complex left half-plane, where I is the identity matrix. The matrix is used to form a Liapunov equation

$$\tilde{A}Z + Z\tilde{A}' = -2BB'$$

whose solution Z is used to compute F through

$$F = B'Z^{+}$$

where + denotes matrix pseudoinverse. It can be shown (ref. 25) that

$$\mathrm{Re}\left(\lambda^{A-BF}\right) = -\beta$$

where λ^{A-BF} is any controllable eigenvalue of the (A,B) system. The option is provided to input β directly or to have β computed internally as

$$\beta = s\left[\max_{i} \left|\mathrm{Re}\left(\lambda_i^{A}\right)\right| + 0.001\right] \qquad (i = 1,2,\ldots,n)$$

where s is an input scale factor and λ_i^{A} are the eigenvalues of A. If the eigenvalue computation in CSTAB fails, β is set to

$$\beta = 2\|A\|$$

using the ℓ_1 matrix norm (see subroutine NORMS). The Liapunov equation for Z can be solved using either subroutine BILIN or subroutine BARSTW. The pseudoinverse of Z is computed through subroutine SNVDEC. Computational parameters for BILIN, BARSTW, and SNVDEC are provided internally through the COMMON block TOL found in subroutine RDTITL.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL CSTAB(A,NA,B,NB,F,NF,IOP,SCLE,DUMMY)

Input arguments:

A, B	Matrices packed by columns in one-dimensional arrays; not destroyed upon normal return
NA, NB	Two-dimensional vectors giving the number of rows and columns of the respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon normal return
IOP	Three-dimensional option vector: IOP(1) = 0 Do not print results. Otherwise Print input, β, F, and eigenvalues of (A-BF). IOP(2) = 0 Do not compute parameter β but use β = SCLE. Otherwise Compute β using s = SCLE. IOP(3) = 0 Use the BARSTW algorithm to solve the Z equation. Otherwise Use BILIN.
SCLE	Parameter used to define β
DUMMY	Vector of working space for computations of dimension at least: $2n^2 + 2(n+1)^2$ if IOP(3) = 0 $6n^2 + 2n$ if IOP(3) $\neq$ 0

Output arguments:

F Matrix packed by columns into a one-dimensional array of dimension at least nr. Upon normal return, F contains $B'Z^+$.

NF Two-dimensional vector holding, upon normal return, the number of rows and columns of F:
NF(1) = NB(2)
NF(2) = NA(1)

COMMON block: TOL

Error messages:

(1) In the β computation section, if the eigenvalue computation for A fails, the message "IN CSTAB, THE SUBROUTINE EIGEN FAILED TO DETERMINE THE ______ EIGENVALUE FOR THE MATRIX A AFTER 30 ITERATIONS" is printed, and the computation continues with $\beta = 2\|A\|$.

(2) If SNVDEC fails to compute the singular values of Z, the message "IN CSTAB, SNVDEC HAS FAILED TO CONVERGE TO THE ______ SINGULAR VALUE AFTER 30 ITERATIONS" is printed, and the program is returned to the calling point.

(3) If a singular value of Z is greater than but close to the ZTEST value used (see SNVDEC and TOL), the message "IN CSTAB, THE MATRIX SUBMITTED TO SNVDEC USING ZTEST = ______ IS CLOSE TO A MATRIX OF LOWER RANK/IF THE ACCURACY IAC IS REDUCED THE RANK MAY ALSO BE REDUCED/CURRENT RANK = ______" is printed along with the singular values of Z after which computation continues.

Field length: 1076 octal words (574 decimal)

Subroutines employed by CSTAB: EQUATE, EIGEN, LNCNT, TRANP, MULT, SCALE, BARSTW, BILIN, SNVDEC, PRNT, SUBT, NORMS

Subroutines employing CSTAB: DSTAB, ASYMREG

Comments: None

Subroutine DSTAB

Description: The purpose of DSTAB is to compute a gain matrix F which, when used in the control law,

$$u = -Fx$$

and applied to the stabilizable linear time-invariant discrete system,

$$x(i+1) = A\,x(i) + B\,u(i)$$

produces a closed-loop response matrix $(A - BF)$ whose eigenvalues lie inside the unit circle of the complex plane. The primary use for DSTAB in ORACLS is to generate a stabilizing gain matrix for initializing the quasilinearization method for solving the discrete steady-state Riccati equation (ref. 10). The matrix $(A - BF)$ computed here has some interesting root-locus properties (ref. 25) which may make the control law applicable in other areas.

Computation follows the method described by Armstrong and Rublein (ref. 12). The matrices A and B are of dimension $n \times n$ and $n \times r$ $(r \leq n)$, respectively. A scalar α is first selected so that

$$0 < \alpha < \min_i\left(1,\ \min_i \left|\lambda_i^A\right|\right) \qquad (i = 1,2,\ldots,n)$$

where $\left|\lambda_i^A\right|$ denotes the nonzero complex moduli of eigenvalues of A. Next, solve the matrix equation,

$$AZA' = \alpha^2 Z + 2BB'$$

for $Z = Z' \geq 0$, and by assuming the controllable eigenvalues of A are nonzero, the stabilizing gain is given by

$$F = B'(Z + BB')^{+}A$$

where + denotes matrix pseudoinverse. If any controllable eigenvalue of A is zero or unknown, it can be made nonzero by applying to the system a prefilter with gain G so that the controllable eigenvalues of $(A - BG)$ are strictly in the complex left half-plane. Such a gain can be computed from CSTAB. Afterward, DSTAB is applied to the $(A - BG, B)$ pair to compute a digital stabilizing gain $\tilde{F}$ with the net stabilizing gain F for the original system given by

$$F = G + \tilde{F}$$

The matrix equation in Z is transformed into

$$\tilde{A}Z + Z\tilde{A}' = \tilde{B}\tilde{B}'$$

with

$$\tilde{A} = (\alpha I + A)^{-1}(A - \alpha I)$$

$$\tilde{B} = 2(\alpha I + A)^{-1}B$$

and solved by subroutine BARSTW. The symbol I denotes an identity matrix.

The option is provided to input α directly or to have α computed internally as

$$\alpha = s \min \left(1, \min_{i} \left|\lambda_i^{A-BG}\right|\right) \qquad (i = 1,2,\ldots,n)$$

where λ_i^{A-BG} are nonzero complex moduli of eigenvalues of $(A - BG)$, $0 < s < 1$ is input scale factor, and G is a gain computed, if requested, internally from CSTAB to cause the controllable eigenvalues of $(A - BG)$ to be nonsingular. The pseudoinverse is computed through SNVDEC. Computational parameters for SNVDEC and BARSTW are provided internally through the COMMON block TOL found in subroutine RDTITL.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL DSTAB(A,NA,B,NB,F,NF,SING,IOP,SCLE,DUMMY)

Input arguments:

A, B	Matrices packed by columns in one-dimensional arrays; not destroyed upon normal return
NA, NB	Two-dimensional vectors giving the number of rows and columns in the respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon normal return

SING Logical variable:

FALSE If the controllable eigenvalues of A are known to be nonzero

TRUE Otherwise, and a gain G will automatically be computed (using CSTAB) before proceeding to the digital algorithm

IOP Two-dimensional option vector:

IOP(1) = 0 Do not print results.

Otherwise Print input, F, α, eigenvalues of $A - BF$, and their magnitudes.

IOP(2) = 0 Do not compute parameter α but use α = SCLE.

Otherwise Compute α by using s = SCLE.

SCLE Parameter used to define α

DUMMY Vector of working space for computations, dimensioned at least $4n^2 + 2(n + 1)^2$

Output arguments:

F Matrix packed by columns into a one-dimensional array of dimension at least nr. Upon normal return, F contains $B'(Z + BB')^+A$.

NF Two-dimensional vector holding, upon normal return, the number of rows and columns in F:

NF(1) = NB(2)

NF(2) = NA(1)

COMMON block: TOL

Error messages:

(1) In the α computation, if the eigenvalue computation for $(A - BG)$ fails, the message "IN DSTAB, THE PROGRAM EIGEN FAILED TO DETERMINE THE ______ EIGENVALUE FOR THE MATRIX A - BG AFTER 30 ITERATIONS" is printed followed by the matrices $(A - BG)$ and G, if SING = TRUE, and the program is returned to the calling point.

(2) If the matrix $A + \alpha I$ (or $(A - BG) + \alpha I$) is found to be singular, the message "IN DSTAB, GELIM HAS FOUND THE MATRIX A + (ALPHA)I (or (A-BG) + (ALPHA)I) SINGULAR" is printed followed by A, G, and α, and the program is returned to the calling point.

(3) If SNVDEC fails to compute the singular values of $(Z + BB')$, the message "IN DSTAB, SNVDEC HAS FAILED TO CONVERGE TO THE ______ SINGULAR VALUE AFTER 30 ITERATIONS" is printed, and the program is returned to the calling point.

(4) If a singular value of (Z + BB') is greater than, but close to the ZTEST value used (see SNVDEC and TOL), the message "IN DSTAB, THE MATRIX SUBMITTED TO SNVDEC, USING ZTEST = ______, IS CLOSE TO A MATRIX OF LOWER RANK/IF THE ACCURACY IAC IS REDUCED THE RANK MAY ALSO BE REDUCED/CURRENT RANK = _____" is printed along with the singular values of (Z + BB') after which computation continues.

(5) If the eigenvalue computation for (A - BF) fails, the message "IN DSTAB, THE PROGRAM EIGEN FAILED TO DETERMINE THE ______ EIGENVALUE FOR THE A-BF MATRIX AFTER 30 ITERATIONS" is printed along with the computed eigenvalues, and the program is returned to the calling point.

Field length: 1605 octal words (901 decimal)

Subroutines employed by DSTAB: CSTAB, MULT, SUBT, EQUATE, EIGEN, GELIM, LNCNT, PRNT, TRANP, MULT, SCALE, BARSTW, ADD, SNVDEC, JUXTC

Subroutine employing DSTAB: ASYMREG

Comments: None

Subroutine DISCREG

Description: The purpose of DISCREG is to solve the time-invariant discrete-time linear optimal output regulator problem with noise-free measurements. Given the digital linear system

$$x(i+1) = A\,x(i) + B\,u(i) + w(i)$$

where $x(0) = x_0$ is given, A and B are constant matrices of dimension $n \times n$ and $n \times r$ $(r \leq n)$, respectively, and $w(i)$ $(i = 0,1,\ldots,N-1)$ is a sequence of uncorrelated zero-mean stochastic variables with variance matrices $V(i)$. Outputs, or controlled variables, are of the form

$$y(i) = H\,x(i)$$

where H is a constant $m \times n$ $(m \leq n)$ matrix. The considered optimal regulator problem is to find the control sequence $u(i)$ which minimizes

$$J = E\left\{ \sum_{i=0}^{N-1} \left[y'(i+1)\,Q\,y(i+1) + u'(i)\,R\,u(i) \right] + x'(N)\,P_N\,x(N) \right\}$$

with $Q = Q' \geq 0$, $P_N = P_N' \geq 0$, and $R = R' > 0$. The symbol E denotes expected value. The solution is given by (ref. 26)

$$\left.\begin{aligned}
u(i) &= -F(i)\,x(i) \\
F(i) &= \left[R + B'\,P(i+1)\,B\right]^{-1} B'\,P(i+1)\,A \\
P(i) &= \phi'(i)\,P(i+1)\,\phi(i) + F'(i)\,R\,F(i) + H'QH \\
\phi(i) &= A - BF(i)
\end{aligned}\right\} \quad (i = 0,1,\ldots,N-1)$$

where $P(N) = P_N$. The minimal value of the criterion function is

$$J = x'(0)\ P(0)\ x(0) + \sum_{j=0}^{N-1} \text{tr}\left[V(j)\ P(j+1)\right]$$

where tr denotes the trace of a matrix.

The program DISCREG repetitively evaluates the solution equations from an input value N to stage zero or until P(i) converges to a steady-state value, whichever occurs first. Numerically, convergence is assumed to have occurred when the improvement in the element of largest magnitude (measured relatively if the magnitude is less than unity, and absolutely otherwise) is less than the parameter RICTCV found in COMMON block CONV of subroutine RDTITL. The program DISCREG does not evaluate the optimal value of the criterion function or print a response trajectory. The subroutine SNVDEC is used to evaluate the matrix inverse in the F(i) equation to allow the option of using nonnegative definite (instead of strictly positive definite) matrices R. Parameters for SNVDEC are set internally by use of the COMMON block TOL of RDTITL.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL DISCREG(A,NA,B,NB,H,NH,Q,NQ,R,NR,F,NF,P,NP,IOP, IDENT,DUMMY)

Input arguments:

A, B, H, Q, R	Matrices packed by columns in one-dimensional arrays. Upon normal return, all except Q are not destroyed.
NA, NB, NH, NQ, NR	Two-dimensional vectors giving the number of rows and columns of the respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon normal return except for NQ = NA
P	Symmetric nonnegative definite matrix packed by columns in a one-dimensional array. On input, P contains the matrix P_N. Afterward, intermediate values of P(i) are stored in the array P.
NP	Two-dimensional vector giving the number of rows and columns of P: NP(1) = Number of rows in P NP(2) = Number of columns in P Not destroyed upon normal return

IOP	Three-dimensional option vector: IOP(1) = 0 Do not print results. Otherwise Print input, and $F(i)$ and $P(i)$ at stage zero or steady state, whichever occurs first. IOP(2) = 0 Do not print at intermediate states between initial and final states. Otherwise Print stage count and values of $F(i)$ and $P(i)$, regardless of printing specified by IOP(1). IOP(3) is terminal stage N for the optimal regulator problem.
IDENT	Logical variable: TRUE If H is an identity matrix FALSE Otherwise For IDENT = TRUE, no data need be input for H, but the related arguments should still appear in the calling sequence.
DUMMY	Vector of working space for computations of dimension at least $4n^2$

Output arguments:

Q	Upon normal return, Q is replaced by H'QH. The array Q must be dimensioned at least n^2.
F	Matrix packed by columns in a one-dimensional array of dimension at least nr. Upon normal return, F contains the value of $F(i)$ at stage zero or the stage at which numerical steady-state convergence occurs.
NF	Two-dimensional matrix giving, upon normal return, the number of rows and columns of F: NF(1) = NB(2) NF(2) = NA(1)
P	Upon normal return, P contains the value of $P(i)$ at stage zero or the stage at which steady-state convergence occurs.

COMMON blocks: TOL, CONV

Error messages:

(1) If SNVDEC fails to compute the singular values of any $[R + B' P(i) B]$, the message "IN DISCREG, SNVDEC HAS FAILED TO CONVERGE TO THE _____ SINGULAR VALUE AFTER 30 ITERATIONS" is printed, and the program is returned to the calling point.

(2) If a singular value of $[R + B' P(i) B]$ is greater than but close to the ZTEST value used (see SNVDEC and TOL), the message "IN DISCREG, THE MATRIX SUBMITTED TO SNVDEC USING ZTEST = ______ IS CLOSE TO A MATRIX OF LOWER RANK/IF THE ACCURACY IAC IS REDUCED THE RANK MAY ALSO BE REDUCED/CURRENT RANK = ______" is printed along with the singular values of $[R + B'P(i)B]$ after which computation continues.

Field length: 1355 octal words (749 decimal)

Subroutines employed by DISCREG: LNCNT, PRNT, MULT, TRANP, EQUATE, SNVDEC, ADD, SUBT, MAXEL

Subroutine employing DISCREG: ASYMREG

Comments: Of course, if $w(i) = 0$ $(i = 0,1,\ldots)$, the algorithm in DISCREG generates the solution to the discrete deterministic optimal linear output regulator problem.

Subroutine CNTNREG

Description: The purpose of CNTNREG is to solve the time-invariant continuous-time linear optimal output regulator problem with noise-free measurements. Given the continuous linear system,

$$\dot{x}(t) = A\,x(t) + B\,u(t) + w(t)$$

where $x(0) = x_0$ is given, A and B are constant matrices of dimension $n \times n$ and $n \times r$ $(r \leq n)$, respectively, and $w(t)$ is zero-mean Gaussian white noise with intensity $V(t)$. Outputs, or controlled variables, are modeled as

$$y(t) = H\,x(t)$$

where H is a constant $m \times m$ $(m \leq n)$ matrix. The considered optimal regulator problem is to find the control function $u(t)$ which minimizes

$$J = E\left\{\int_0^{t_1} \left[y'(t)\,Q\,y(t) + u'(t)\,R\,u(t)\right] dt + x'(t_1)\,P_1\,x(t_1)\right\}$$

where $Q = Q' \geq 0$, $P_1 = P_1' \geq 0$, and $R = R' > 0$. The symbol E denotes expected value. The solution is given by reference 4 as

$$u(t) = -F(t)\,x(t)$$

$$F(t) = R^{-1}B'\,P(t)$$

$$\frac{-dP(t)}{dt} = H'QH + A'\,P(t) + P(t)\,A - P(t)\,BR^{-1}B'\,P(t)$$

where $P(t_1) = P_1$. The minimal value of the criterion function is

$$x'(0)\,P(0)\,x(0) + \int_0^{t_1} \mathrm{tr}\left[P(t)\,V(t)\right] dt$$

where tr denotes the trace of a matrix.

The computational algorithm used in CNTNREG to solve the Riccati equation is that due to Vaughan (ref. 8) as described by Kwakernaak and Sivan (ref. 4). From

$$Z = \begin{bmatrix} A & -BR^{-1}B' \\ & \\ -H'QH & -A' \end{bmatrix}$$

The Z matrix has the property that if λ is an eigenvalue, then $-\lambda$ is also an eigenvalue. By assuming that Z has linearly independent eigenvectors, there exists a matrix W such that

$$Z = W\begin{pmatrix} \Lambda & 0 \\ & \\ 0 & -\Lambda \end{pmatrix}W^{-1}$$

where Λ is a diagonal matrix constructed as follows. If an eigenvalue λ of Z has $\mathrm{Re}(\lambda) > 0$, it is a diagonal element of Λ, and $-\lambda$ is automatically placed in $-\Lambda$. If $\mathrm{Re}(\lambda) = 0$, one of the pair $(\lambda,-\lambda)$ is arbitrarily assigned to Λ and the other to $-\Lambda$. The matrix W is composed of eigenvectors of Z; the ith column vector of W is the eigenvector of Z corresponding to the eigenvalue in the ith diagonal position of $\mathrm{diag}(\Lambda,-\Lambda)$. In practice, it is noted that if λ is a complex eigenvalue of Z with eigenvector ν, then their complex conjugates $\bar{\lambda}$ and $\bar{\nu}$ are also an eigenpair and $[\mathrm{Re}(\nu),\mathrm{Im}(\nu)]$ is placed in W instead of $(\nu,\bar{\nu})$ in order to avoid complex arithmetic. This has the effect of making Λ block diagonal where every $(\lambda,\bar{\lambda})$ pair go together to form a 2×2 real matrix placed at the former $(\lambda,\bar{\lambda})$ entry in Λ. Next partition the $2n \times 2n$ matrix W into four $n \times n$ blocks as

$$W = \begin{bmatrix} W_{11} & W_{12} \\ & \\ W_{21} & W_{22} \end{bmatrix}$$

The solution P(t) of the Riccati equation can be written as

$$P(t) = \left[W_{22} + W_{21}\,G(t_1 - t)\right]\left[W_{12} + W_{11}\,G(t_1 - t)\right]^{-1}$$

with

$$G(t) = e^{-\Lambda t}Se^{-\Lambda t}$$

and

$$S = -(W_{21} - P_1W_{11})^{-1}(W_{22} - P_1W_{12})$$

If Λ is composed of eigenvalues with strictly positive real parts, then the steady-state solution to the Riccati equation is given directly as

$$\lim_{t_1 \to \infty} P(t) = W_{22} W_{12}^{-1}$$

The program CNTNREG evaluates the solution equations from an input value t_1 to time zero or until $P(t)$ converges to a steady-state value, whichever occurs first. Numerically, convergence is assumed to have occurred when the improvement in the element of largest magnitude (measured relatively if the magnitude is less than unity, and absolutely otherwise) is less than the parameter RICTCV found in the COMMON block CONV of subroutine RDTITL. CNTNREG does not evaluate the optimal value of the criterion function or print a response trajectory. Options are provided to compute directly the steady-state solution and to compute Z, W, S, Λ, and $e^{-\Lambda t}$ only without computing any of the $P(t)$ values.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL CNTNREG(A,NA,B,NB,H,NH,Q,NQ,R,NR,Z,W,LAMBDA,S,F,NF, P,NP,T,IOP,IDENT,DUMMY)

Input arguments:

A, B, H, Q, R	Matrices packed by columns in one-dimensional arrays. Upon normal return, all except Q are not destroyed.
NA, NB, NH, NQ, NR	Two-dimensional vectors giving the number of rows and columns in the respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon normal return except for NQ replaced by NA
P	Symmetric nonnegative definite matrix packed in a one-dimensional array. On input, P contains the matrix P_1. Afterward, intermediate values of $P(t)$ are stored in the array P.
NP	Two-dimensional vector giving the number of rows and columns in P: NP(1) = Number of rows of P NP(2) = Number of columns of P Not destroyed upon normal return

T	Two-dimensional vector: T(1) = Final time t_1 T(2) = Increment of time for transient solution computation The final time t_1 must be expressed as an integer multiple of T(2). The vector T is not required if only the steady-state solution of P(t) is required but must still appear as an argument of the calling sequence; not destroyed upon normal return. Set T(1) negative if only Z, W, S, Λ, and $e^{-\Lambda T(2)}$ are required without any P(t) computation.
IOP	Three-dimensional option vector: IOP(1) = 0 Do not print results, nor compute $W^{-1}ZW$. Otherwise Print input, Z, λ^{Z}, W, Λ, $W^{-1}ZW$, W_{11}, W_{12}, W_{21}, W_{22}, $R^{-1}B'$, $e^{-\Lambda T(2)}$ and values of F and P at time zero or steady state, whichever comes first. IOP(2) = 0 Do not print at intermediate times (multiples of T(2)) between T(1) and zero or steady state. Otherwise Regardless of the printing specified by IOP(1), print values of P and F at intermediate times. The parameter IOP(2) is not required if only a steady-state solution is required. IOP(3) = 0 Compute transient solutions P(t) and F(t). Otherwise Compute only steady-state values of P and F.
IDENT	Logical variable: TRUE If H is an identity matrix FALSE Otherwise For IDENT = TRUE, no data need be input in H but the related arguments should still appear in the calling sequence.
DUMMY	Vector of working space for computations dimensioned at least: $8n^2 + 18n + nr$ for IOP(3) $\neq 0$ $9n^2 + 17n + nr$ for IOP(3) $= 0$

Output arguments:

Q	Upon normal return, Q is replaced by H'QH. The array Q must be dimensioned at least n^2.

Z, W, LAMBDA, S	Matrices packed by columns in one-dimensional arrays dimensioned at least $4n^2$, $4n^2$, n^2, and n^2, respectively. Upon normal return, these arrays contain their theoretical counterparts. The array LAMBDA (Λ) is declared real.
F	Matrix packed by columns in a one-dimensional array of dimension at least nr. Upon normal return, F contains the value of F(t) at time zero or the time at which steady-state convergence numerically occurs.
NF	Two-dimensional matrix giving, upon normal return, the number of rows and columns of F: NF(1) = NB(2) NF(2) = NA(1)
P	Upon normal return, P contains the value of P(t) at time zero or the time at which steady-state convergence numerically occurs.
DUMMY	Upon normal return, the first nr elements contain the matrix $R^{-1}B'$, the next $4n^2$ contain the submatrices W_{11}, W_{21}, W_{12}, and W_{22} stored in block column form and, if a transient solution is computed, the next n^2 contain the matrix $e^{-\Lambda T(2)}$.

COMMON block: CONV

Error messages:

(1) If the computation of $R^{-1}B'$ fails, the message "IN CNTNREG, THE SUBROUTINE SYMPDS HAS FOUND THE MATRIX R NOT SYMMETRIC POSITIVE DEFINITE" is printed, and the program is returned to the calling point.

(2) If the eigenvalue/eigenvector computation for Z fails, either the message "IN CNTNREG, THE ______ EIGENVALUE OF Z HAS NOT BEEN FOUND AFTER 30 ITERATIONS" or "IN CNTNREG, EIGEN FAILED TO COMPUTE THE ______ EIGENVECTOR OF Z" is printed, and the program is returned to the calling point.

(3) If the computation for $W^{-1}ZW$ fails, the message "IN CNTNREG, GELIM HAS FOUND THE REORDERED MATRIX W TO BE SINGULAR" is printed, and computation continues.

(4) If the computation of S for the transient solution fails, the message "IN CNTNREG, GELIM HAS FOUND THE MATRIX W21 - P1X W11 TO BE SINGULAR" is printed, and the program is returned to the calling point.

(5) If the computation of W_{12}^{-1} fails in the steady-state case, the message "IN CNTNREG, GELIM HAS FOUND THE MATRIX W12 TO BE SINGULAR" is printed, and the program is returned to the calling point.

(6) If at any time, computation of the matrix inverse in the $P(t)$ computation fails, the message "IN CNTNREG AT TIME ______ P CANNOT BE COMPUTED DUE TO MATRIX SINGULARITY IN GELIM" is printed, and the program is returned to the calling point.

Field length: 3046 octal words (1574 decimal)

Subroutines employed by CNTNREG: EQUATE, TRANP, SYMPDS, SCALE, MULT, JUXTR, EIGEN, GELIM, PRNT, SUBT, EXPSER, MAXEL, LNCNT, NULL

Subroutine employing CNTNREG: ASYMREG

Comments: Of course, if $w(t) = 0$ for all t, then the algorithm of CNTNREG produces the solution to the deterministic optimal linear output regulator problem. In this case, the system response $x(t)$ to the steady-state optimal control law

$$u(t) = -R^{-1}B'W_{22}W_{12}^{-1}\,x(t)$$

is given by

$$x(t) = W_{12}e^{-\Lambda t}W_{12}^{-1}x_0$$

Subroutine RICTNWT

Description: The purpose of RICTNWT is to solve either the continuous or discrete steady-state Riccati equation by the Newton algorithms described by Kleinman (ref. 9) and Hewer (ref. 10).

For the continuous case, the algebraic Riccati equation is

$$PA + A'P + H'QH - PBR^{-1}B'P = 0$$

where the constant matrices A, B, H, $Q = Q' \geq 0$, and $R = R' \geq 0$ are of dimension $n \times n$, $n \times r$ $(r \leq n)$, $m \times n$ $(m \leq n)$, $m \times m$, and $r \times r$, respectively. Applying Newton's equation-solving algorithm (ref. 3) to solve the continuous P equation leads to the sequence

$$\left.\begin{aligned} 0 &= \phi_k{}'P_k + P_k\phi_k + H'QH + F_k{}'RF_k \\ \phi_k &= A - BF_k \\ F_k &= R^{-1}B'P_{k-1} \end{aligned}\right\} \qquad (k = 1,2,\ldots)$$

Kleinman (ref. 9) and Sandell (ref. 27) established that if the (A,B) pair is stabilizable and the (A,D) pair (with D such that $D'D = H'QH$) is detectable (ref. 4), the sequence $P_k = P_k{}' \geq 0$ $(k = 0,1,\ldots)$ converges to the correct $P = P' \geq 0$ when F_1 is chosen so that $(A - BF_1)$ is asymptotically stable in the continuous sense. In RICTNWT, the option is provided to solve the continuous Liapunov equation for P_k by either of the BILIN or BARSTW subroutines.

For the discrete case, the steady-state Riccati equation can be written as

$$P = \phi'P\phi + F'RF + H'QH$$

with

$$\phi = A - BF$$

$$F = (R + B'PB)^{-1}B'PA$$

and A, B, H, Q, and R as previously defined. Applying Newton's algorithm gives the sequence

$$\left.\begin{aligned} P_k &= \Phi_k' P_k \Phi_k + F_k' R F_k + H'QH \\ F_k &= (R + B'P_{k-1}B)^{-1}B'P_{k-1}A \\ \Phi_k &= A - BF_k \end{aligned}\right\} \qquad (k = 1,2,\ldots)$$

It can be established that if the (A,B) pair is stabilizable and the (A,D) pair (with D such that $D'D = H'QH$) is detectable, the sequence $P_k = P_k' \geq 0$ $(k = 0,1,\ldots)$ converges to the correct $P = P' \geq 0$ when F_1 is chosen so that $(A - BF_1)$ is asymptotically stable in the discrete sense. In RICTNWT, the discrete Liapunov equation in P_k is solved by subroutine SUM.

In both the continuous and discrete cases, RICTNWT assumes that an input F_1 is provided such that $(A - F_1)$ is asymptotically stable in the appropriate sense. If A is already stable, $F_1 = 0$ suffices. Otherwise, the subroutines CSTAB and DSTAB are available for computing initializing F_1 matrices. RICTNWT repetitively evaluates the solution sequence equations for up to 100 iterations or until P_k converges to P. Numerically, convergence is assumed to have occurred when the improvement in the element of largest magnitude (measured relatively if the magnitude is less than unity, and absolutely otherwise) is less than the parameter RICTCV found in COMMON block CONV of subroutine RDTITL. Parameters for use in subroutine BARSTW are set internally through the COMMON block TOL of RDTITL.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL RICTNWT(A,NA,B,NB,H,NH,Q,NQ,R,NR,F,NF,P,NP,IOP,IDENT, DISC,FNULL,DUMMY)

Input arguments:

A, B, H, Q, R	Matrices packed by columns into one-dimensional arrays. Upon normal return, all except Q are not destroyed.
NA, NB, NH, NQ, NR	Two-dimensional vectors giving the number of rows and columns of the respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon normal return except for NQ replaced by NA
F	Matrix packed by columns into a one-dimensional array of size at least nr. If the matrix A is not asymptotically stable, the input F causes (A - BF) to be asymptotically stable. For A asymptotically stable, no data for F are required.

NF	Two-dimensional vector giving the number of rows and columns of F: NF(1) = r NF(2) = n Not required as input if data for F are not input
IOP	Three-dimensional option vector: IOP(1) = 0 Do not print results. Otherwise Print input data, H'QH, and final values of F and P. IOP(2) = 0 Do not print at each iteration. Otherwise Regardless of printing specified by IOP(1), print iteration count and value of P. IOP(3) = 0 Use subroutine BARSTW to solve the continuous Liapunov equation. Otherwise Use subroutine BILIN. IOP(3) is not required if the discrete Riccati equation is to be solved.
IDENT	Logical variable: TRUE If H is an identity matrix FALSE Otherwise If H is an identity matrix, no data need be input for H, but H and NH must still appear as arguments of the calling sequence.
DISC	Logical variable: TRUE If the digital version is solved FALSE For the continuous version
FNULL	Logical variable: TRUE If F = 0 FALSE Otherwise
DUMMY	Vector of working space for computations dimensioned at least: $5n^2$ if DISC = TRUE $5n^2 + n(r + 4) + 2$ if DISC = FALSE and IOP(3) = 0 $7n^2 + n(r + 2)$ if DISC = FALSE and IOP(3) $\neq$ 0

Output arguments:

Q	Upon normal return, Q is replaced by H'QH. Q must be dimensioned at least n^2.
F	The value of F_k at the stage k of return. If convergence occurs, F contains the steady-state gain matrix.

P	Matrix packed by columns in a one-dimensional array of dimension at least n^2; the value of P_k at the stage k of return. If convergence occurs, P contains the steady-state Riccati equation solution.
NP, NF	Two-dimensional vectors giving the number of rows and columns in P and F: upon normal return, NP = NA NF(1) = r NF(2) = n

COMMON blocks: TOL, CONV

Error messages:

(1) If the computation of either R^{-1} or $(R + B'P_kB)^{-1}$ $(k = 0,1,\ldots)$ fails, the message "IN RICTNWT, A MATRIX WHICH IS NOT SYMMETRIC POSITIVE DEFINITE HAS BEEN SUBMITTED TO SYMPDS" is printed, and the program is returned to the calling point.

(2) If the iteration count exceeds 100, the message "THE SUBROUTINE RICTNWT HAS EXCEEDED 100 ITERATIONS WITHOUT CONVERGENCE" is printed, IOP(1) is set to 1, P and F are printed, and the program is returned to the calling point.

Field length: 2415 octal words (1293 decimal)

Subroutines employed by RICTNWT: LNCNT, PRNT, MULT, TRANP, EQUATE, SYMPDS, SCALE, SUBT, ADD, BARSTW, BILIN, MAXEL, SUM

Subroutine employing RICTNWT: ASYMREG

Comments: None

Subroutine ASYMREG

Description: The purpose of ASYMREG is to solve either the continuous or discrete time-invariant asymptotic linear optimal output regulator problem with noise-free measurements. For the continuous-time deterministic case, the control function u(t) is chosen to minimize the criterion function,

$$J = \lim_{t_1 \to \infty} \int_0^{t_1} \left[y'(t)\ Q\ y(t) + u'(t)\ R\ u(t) \right] dt$$

subject to

$$\dot{x}(t) = A\ x(t) + B\ u(t)$$

$$y(t) = H\ x(t)$$

where $x(0) = x_0$ is given and A, B, H, $Q = Q' \geq 0$, and $R = R' > 0$ are constant matrices of dimension $n \times n$, $n \times r\ (r \leq n)$, $m \times n\ (m \leq n)$, $m \times m$, and $r \times r$, respectively. If the (A,B) pair is stabilizable and the (A,D) pair (with $D'D = H'QH$) is detectable, a solution to the optimal control problem exists and is given by

$$u(t) = -F\ x(t)$$

where

$$F = R^{-1}B'P$$

$$PA + A'P + H'QH - PBR^{-1}B'P = 0$$

with the criterion function taking the value

$$x'(0)\ P\ x(0)$$

The same control law satisfies the stochastic continuous version of the problem (ref. 4) in which the criterion function is

$$J = \lim_{t \to \infty} E\left[y'(t)\ Q\ y(t) + u'(t)\ R\ u(t) \right]$$

where E is expected value, and the state equation is

$$\dot{x}(t) = A\, x(t) + B\, u(t) + w(t)$$

where $w(t)$ is a zero-mean Gaussian white noise with intensity V. The optimal value of the stochastic criterion function is

$$\mathrm{trace}\ (PV)$$

For the deterministic discrete-time case, the control function $u(i)$ $(i = 0,1,\ldots)$ is chosen to minimize the criterion function

$$J = \lim_{N\to\infty} \left\{ \sum_{i=0}^{N-1} \left[y'(i+1)\, Q\, y(i+1) + u'(i)\, R\, u(i) \right] \right\}$$

subject to

$$x(i+1) = A\, x(i) + B\, u(i)$$

$$y(i) = H\, x(i)$$

where $x(0) = x_0$ is given and A, B, H, Q, and R are as previously defined. If the (A,B) pair is stabilizable and the (A,D) pair (with $D'D = H'QH$) is detectable, a solution to the optimal control exists and is given by

$$u(i) = -F\, x(i)$$

where

$$F = (R + B'PB)^{-1}B'PA$$

$$P = \phi'P\phi + F'RF + H'QH$$

$$\phi = A - BF$$

with the criterion function taking the value

$$x'(0)\ P\ x(0)$$

The same control law satisfies the stochastic discrete-time version of the problem (ref. 4) in which the criterion function is

$$J = \lim_{N\to\infty} \frac{1}{N} E\left\{ \sum_{i=0}^{N-1} \left[y'(i+1)\ Q\ y(i+1) + u'(i)\ R\ u(i) \right] \right\}$$

and the state equation is

$$x(i+1) = A\ x(i) + B\ u(i) + w(i)$$

where $w(i)$ $(i = 0,1,\ldots)$ is a sequence of zero-mean stochastic variables with variance matrix V. The optimal value of the stochastic criterion function is

$$\text{trace}\ (PV)$$

ASYMREG does not evaluate the optimal values of the performance criteria. Therefore, no V data are input. The option of solving the appropriate steady-state Riccati equation using either of the subroutines DISCREG, CNTNREG, or RICTNWT is provided. In addition, the residual error in the Riccati equation, the eigenvalues of P, the closed-loop response matrix $(A - BF)$, and the eigenvalues of $(A - BF)$ can be computed. If RICTNWT is selected and if the matrix A is not asymptotically stable or it is unknown whether A is asymptotically stable, the option is provided to test the relative stability of A and compute, if necessary, a stabilizing gain to initialize the Newton process.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL ASYMREG(A,NA,B,NB,H,NH,Q,NQ,R,NR,F,NF,P,NP,IDENT,DISC, NEWT,STABLE,FNULL,ALPHA,IOP,DUMMY)

Input arguments:

A, B, H, Q, R	Matrices packed by columns into one-dimensional arrays. Upon normal return, all except Q are not destroyed.

CONTENTS

SUMMARY 1 1/A9

INTRODUCTION 1 1/A9

OVERVIEW OF ORACLS 2 1/A10

PRIMARY SUBROUTINES FOR INPUT-OUTPUT 5 1/A13
Subroutine RDTITL[1] 5 1/A13
(Input Hollerith data; define COMMON block data)
Subroutine LNCNT 6 1/A14
(Accumulate line count; page output)
Subroutine READ 7 1/B1
(Input numerical data)
Subroutine PRNT 8 1/B2
(Output numerical data)

PRIMARY SUBROUTINES FOR VECTOR-MATRIX OPERATIONS 10 1/B4
Subroutine EQUATE 10 1/B4
(Equating matrices, A = B)
Subroutine TRANP 11 1/B5
(Matrix transpose, A')
Subroutine SCALE 12 1/B6
(Scalar multiplication)
Subroutine UNITY 13 1/B7
(Generate identity matrix)
Subroutine NULL 14 1/B8
(Generate null matrix)
Subroutine TRCE 15 1/B9
(Trace of matrix)
Subroutine ADD 16 1/B10
(Matrix addition)
Subroutine SUBT 17 1/B11
(Matrix subtraction)
Subroutine MULT 18 1/B12
(Matrix multiplication)
Subroutine MAXEL 19 1/B13
(Maximum of absolute values of matrix elements)
Subroutine NORMS 20 1/B14
(Compute ℓ_1, ℓ_2, and ℓ_∞ matrix norms)
Subroutine JUXTC 22 1/C2
(Matrix juxtaposition by columns)
Subroutine JUXTR 23 1/C3
(Matrix juxtaposition by rows)

---

[1] Required of all user-provided driver programs.

PRIMARY SUBROUTINES FOR ANALYSIS OF CONSTANT LINEAR SYSTEMS 24 1/C4
Subroutine FACTOR 24 1/C4
(Factor nonnegative definite matrix)
Subroutine EIGEN 26 1/C6
(Eigenvalue-eigenvector computation)
Subroutine SYMPDS 28 1/C8
(Solve $AX = B$, $A = A' > 0$)
Subroutine GELIM 30 1/C10
(Solve $AX = B$, A nonsingular)
Subroutine SNVDEC 32 1/C12
(Singular value decomposition)
Subroutine SUM 35 1/D1
(Solve discrete Liapunov equation)
Subroutine BILIN 37 1/D3
(Solve continuous Liapunov equation)
Subroutine BARSTW 41 1/D7
(Solve general equation $AX + XB = C$)
Subroutine TESTSTA 43 1/D9
(Test matrix for relative stability)
Subroutine EXPSER 45 1/D11
(Matrix exponential by series method)
Subroutine EXPADE 47 1/D13
(Matrix exponential by Padé method)
Subroutine EXPINT 49 1/E1
(Matrix exponential and integral)
Subroutine VARANCE 51 1/E3
(Steady-state variance of linear system)
Subroutine CTROL 54 1/E6
(Controllability matrix)
Subroutine TRANSIT 57 1/E9
(Transient response of forced linear system)

PRIMARY SUBROUTINES FOR IMPLEMENTING LQG CONTROL LAW DESIGN 60 1/E12
Subroutine SAMPL 60 1/E12
(Evaluate sampled data regulator coefficients)
Subroutine PREFIL 64 1/F2
(Eliminate performance index cross-product terms)
Subroutine CSTAB 67 1/F5
(Stabilize continuous systems)
Subroutine DSTAB 70 1/F8
(Stabilize discrete systems)
Subroutine DISCREG 74 1/F12
(Digital transient quadratic regulator)
Subroutine CNTNREG 78 1/G2
(Continuous transient quadratic regulator)
Subroutine RICTNWT 84 1/G8
(Riccati solution by Newton's method)
Subroutine ASYMREG 88 1/G12
(Asymptotic quadratic regulator)
Subroutine ASYMFIL 94 2/A7
(Asymptotic Kalman-Bucy filter)

  Subroutine EXPMDFL 100 2/A13
    (Explicit model following)
  Subroutine IMPMDFL 108 2/B7
    (Implicit model following)

SUPPORTING SUBROUTINES 115 2/B14
  Subroutine READ1 . . . . 115 2/B14
    (Read single matrix)
  Subroutine BALANC 116 2/C1
    (Supports eigenvalue/eigenvector computation)
  Subroutine ELMHES 117 2/C2
    (Supports eigenvalue/eigenvector computation)
  Subroutine HQR 118 2/C3
    (Supports eigenvalue/eigenvector computation)
  Subroutine INVIT 120 2/C5
    (Supports eigenvalue/eigenvector computation)
  Subroutine ELMBAK 122 2/C7
    (Supports eigenvalue/eigenvector computation)
  Subroutine BALBAK 123 2/C8
    (Supports eigenvalue/eigenvector computation)
  Subroutine DETFAC 124 2/C9
    (LU factorization of real square matrix)
  Subroutine AXPXB 126 2/C11
    (Solve AX + XB = C)
  Subroutine SHRSLV 129 2/C14
    (Supports BARSTW)
  Subroutine ATXPXA 130 2/D1
    (Solve A'X + XA = C, C = C')
  Subroutine SYMSLV 132 2/D3
    (Supports BARSTW)
  Subroutine HSHLDR 133 2/D4
    (Supports BARSTW)
  Subroutine BCKMLT 134 2/D5
    (Supports BARSTW)
  Subroutine SCHUR 135 2/D6
    (Supports BARSTW)
  Subroutine SYSSLV 137 2/D8
    (Supports BARSTW)
  Subroutine GAUSEL 138 2/D9
    (Supports EXPADE)

EXAMPLE COMPUTATIONS 139 2/D10
  Example 1 - Optimal Transient Regulator 140 2/D11
  Example 2 - Optimal Sampled-Data Regulator 152 2/F5
  Example 3 - Model Following 163 2/G11
  Example 4 - Kalman-Bucy Filter 183 3/C11

CONCLUDING REMARKS 190 3/D9

REFERENCES 191 3/D10

NA, NB, NH, NQ, NR	Two-dimensional vectors giving the number of rows and columns of the respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon normal return except for NQ replaced by NA
F, NF, P, NP	F and P are matrices packed by columns into one-dimensional arrays of dimension at least nr and n^2, respectively. NF and NP are the corresponding two-dimensional vectors giving the number of rows and columns of P and F Any input requirements depend on and are specified by whether DISCREG, CNTNREG, or RICTNWT is employed.
IDENT	Logical variable: TRUE If H is an identity matrix FALSE Otherwise If H is an identity matrix, no data need be input for H, but H and NH must still appear as arguments of the calling sequence.
DISC	Logical variables: TRUE If the digital version is solved FALSE For the continuous version
NEWT	Logical variable: TRUE Newton's method (RICTNWT) is used to solve the appropriate Riccati equation. FALSE Either CNTNREG or DISCREG is used depending upon the value of DISC.
STABLE	Logical variable. Data for STABLE are not required if NEWT = FALSE, but STABLE must still appear as an argument of the calling sequence. When NEWT = TRUE: STABLE = TRUE If it is known that the matrix (A - BF) computed from input data is stable relative to an input parameter ALPHA. STABLE = FALSE The matrix (A - BF) is evaluated and tested for stability relative to ALPHA using subroutine TESTSTA. If a stabilizing gain is required, it is computed from subroutine DSTAB or CSTAB.
FNULL	Logical variable; data for FNULL are not required if NEWT = FALSE, but FNULL must still appear as an argument of the calling sequence. When NEWT = TRUE: FNULL = TRUE If the input F is a null matrix FNULL = FALSE Otherwise

ALPHA Scalar variable. ALPHA is not required if NEWT = FALSE or STABLE = TRUE. Otherwise, ALPHA is used in the asymptotic stability test of (A - BF) from input data using subroutine TESTSTA.

IOP Five-dimensional option vector:
IOP(1), IOP(2), and IOP(3) are the first three elements of the IOP vector in DISCREG, CNTNREG, or RICTNWT. If CNTNREG is selected, IOP(3) is set internally to a nonzero value.

IOP(4) = 0	Do not compute the Riccati equation residual.
Otherwise	Compute the residual and print.
IOP(5) = 0	Compute but do not print the eigenvalues of P, the matrix (A - BF), and the eigenvalues of (A - BF).
Otherwise	Print these data after computation.

DUMMY Vector of working space for computations dimensioned at least:

DISC	NEWT		
TRUE	TRUE	$6n^2 + 4n + 2$	(STABLE = FALSE)
		$5n^2$	(STABLE = TRUE)
TRUE	FALSE	$4n^2$	
FALSE	FALSE	$17n^2 + n(r + 18)$	
FALSE	TRUE	$8n^2 + n(r + 1)$	(BILIN)
		$5n^2 + n(r + 4) + 2$	(BARSTW)

Output arguments:

Q Upon normal return, Q is replaced by H'QH. The dimension of Q must be at least n^2.

F Upon normal return, F contains the steady-state gain matrix for either the continuous or discrete problem.

NF Two-dimensional vector giving, upon normal return, the number of rows and columns of F:
NF(1) = r
NF(2) = n

P Upon normal return, P contains the steady-state solution for either the continuous or discrete Riccati equation.

NP Two-dimensional vector giving the number of rows and columns of P. Upon normal return, NP = NA.

STABLE Upon normal return, STABLE = TRUE.

DUMMY Upon normal return, the first n elements of DUMMY contain the eigenvalues of P, the next n^2 contain the matrix (A - BF) for steady-state F, and the next $2n$ contain the eigenvalues of (A - BF) stored as an $n \times 2$ matrix with the real parts as first column and imaginary parts as the second. All matrices are packed by columns into one-dimensional arrays.

COMMON blocks: None

Error messages:

(1) If the stabilizing gain computation for the continuous system fails, the message "IN ASYMREG, CSTAB HAS FAILED TO FIND A STABILIZING GAIN MATRIX (F) RELATIVE TO / ALPHA = _____" is printed, and the program is returned to the calling point.

(2) If the stabilizing gain computation for the discrete system fails, the message "IN ASYMREG, DSTAB HAS FAILED TO FIND A STABILIZING GAIN MATRIX (F) RELATIVE TO / ALPHA = _____" is printed, and the program is returned to the calling point.

(3) If the eigenvalue computation for P fails, the message "IN ASYMREG, THE _____ EIGENVALUE OF P HAS NOT BEEN COMPUTED AFTER 30 ITERATIONS" is printed, and the program continues with the computation of (A - BF) and its eigenvalues.

(4) If the eigenvalue computation for (A - BF) fails, the message "IN ASYMREG, THE _____ EIGENVALUE OF A-BF HAS NOT BEEN COMPUTED AFTER 30 ITERATIONS" is printed, and a return is made to the calling point if no printing is required. Otherwise, the available information is printed before return.

Field length: 1756 octal words (1006 decimal)

Subroutines employed by ASYMREG: MULT, SUBT, TESTSTA, CSTAB, SCALE, DISCREG, RICTNWT, TRANP, ADD, EQUATE, EIGEN, JUXTC, LNCNT, PRNT, CNTNREG, DSTAB

Subroutines employing ASYMREG: ASYMFIL, EXPMDFL, IMPMDFL

Comments: When using DISCREG to solve the steady-state discrete Riccati equation, the entry IOP(3) = N must be set sufficiently large for steady-state convergence to occur.

Subroutine ASYMFIL

Description: The purpose of ASYMFIL is to solve either the continuous or discrete time-invariant asymptotic optimal Kalman-Bucy filter problem (ref. 4).

For the continuous case, the state equation is given by

$$\dot{x}(t) = A\ x(t) + G\ \tilde{n}(t)$$

with output

$$y(t) = H\ x(t) + \tilde{m}(t)$$

where A, G, and H are constant matrices of dimension $n \times n$, $n \times m$ $(m \leq n)$, and $r \times n$ $(r \leq n)$, respectively. The process noise $\tilde{n}(t)$ is a zero-mean Gaussian white-noise process with intensity for the covariance given by $Q = Q' \geq 0$. The measurement noise $\tilde{m}(t)$ is also a zero-mean Gaussian white-noise process with intensity matrix $R = R' > 0$. The processes $\tilde{n}(t)$ and $\tilde{m}(t)$ along with the Gaussian $x(t_0)$ are mutually uncorrelated. The optimal filter problem is to construct an estimate $\hat{x}(t)$ of $x(t)$ operating over $\left[t_0,t\right]$ such that the quantity,

$$J = \lim_{t_0 \to -\infty} E\left[e'(t)\ W\ e(t)\right]$$

is minimized where E denotes expected value and

$$e(t) = x(t) - \hat{x}(t)$$

and the constant $n \times n$ matrix satisfies

$$W = W' \geq 0$$

When the pair (A',H') is stabilizable and the (A',D') pair (where $DD' = GQG'$) is detectable, the solution to the asymptotic optimal observer problem exists and $\hat{x}(t)$ is given by

$$\dot{\hat{x}}(t) = A\ \hat{x}(t) + F\left[y(t) - H\ \hat{x}(t)\right]$$

where

$$\hat{x}(t_0) = E\left[x(t_0)\right]$$

with filter gain

$$F = PH'R^{-1}$$

and P satisfying

$$0 = AP + PA' + GQG' - PH'R^{-1}HP$$

The matrix $P = P' \geq 0$ represents the (constant) steady-state variance matrix of the reconstruction error $e(t)$; that is,

$$\lim_{t_0 \to -\infty} E\left[e(t)\, e'(t)\right] = P$$

Also,

$$J = \lim_{t_0 \to -\infty} E\left[e'(t)\, W\, e(t)\right] = \text{trace}\ (PW)$$

For the discrete case,

$$x(i+1) = A\, x(i) + G\, \tilde{n}(i)$$

with output

$$y(i) = H\, x(i) + \tilde{m}(i)$$

where A, G, and H are as previously defined. The process noise $\tilde{n}(i)$ $(i = i_0, i_0+1, \ldots)$ is a sequence of zero-mean Gaussian white-noise stochastic variables with variance matrix $Q = Q' \geq 0$. The measurement noise $\tilde{m}(i)$ $(i = i_0, i_0+1, \ldots)$ is also a zero-mean Gaussian white-noise (ref. 4) sequence with variance $R = R' > 0$. The processes $\tilde{n}(i)$ and $\tilde{m}(i)$ along with the Gaussian $x(i_0)$ are mutually uncorrelated. The optimal estimator problem considered here (technically a prediction problem) is to construct an estimate $\hat{x}(i)$ of $x(i)$ from knowledge of $y(i_0), y(i_0+1), \ldots, y(i-1)$ such that the quantity,

$$J = \lim_{i_0 \to -\infty} E\left[e'(i)\ W\ e(i)\right]$$

is minimized, where

$$e(i) = x(i) - \hat{x}(i)$$

with W as previously defined. When the (A',H') pair is stabilizable and the (A',D') pair (with $DD' = GQG'$) is detectable, the solution to the asymptotic optimal observer problem exists and $\hat{x}(i)$ is given by

$$\hat{x}(i+1) = A\ \hat{x}(i) + F\left[y(i) - H\ \hat{x}(i)\right]$$

where

$$\hat{x}(i_0) = E\left[x(i_0)\right]$$

with filter gain

$$F = APH'(R + HPH')^{-1}$$

and P satisfying

$$P = \Phi P \Phi' + FRF' + GQG'$$

for

$$\Phi = A - FH$$

The matrix $P = P' \geq 0$ represents the (constant) steady-state variance matrix of the reconstruction error $e(i)$; that is,

$$\lim_{i_0 \to -\infty} E\left[e(i)\ e'(i)\right] = P$$

Also,

$$J = \lim_{i_0 \to -\infty} E\left[e'(i)\ W\ e(i)\right] = \text{trace }(PW)$$

Computation for both the discrete and continuous versions of the foregoing optimal filter problems is performed using duality theory (ref. 4) and the regulator subroutine ASYMREG. Thus, the user has the option of solving the appropriate steady-state covariance equations in P by either of the subroutines DISCREG, CNTNREG, or RICTNWT. No computations are performed which involve the matrix W; hence, no data for W are required.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL ASYMFIL(A,NA,G,NG,H,NH,Q,NQ,R,NR,F,NF,P,NP,IDENT,DISC, NEWT,STABLE,FNULL,ALPHA,IOP,DUMMY)

Input arguments:

A, G, H, Q, R	Matrices packed by columns in one-dimensional arrays. Upon normal return, all except Q are not destroyed.
NA, NG, NH, NQ, NR	Two-dimensional vectors giving the number of rows and columns of respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon normal return except for NQ replaced by NA
F, NF, P, NP	F and P are matrices packed by columns into one-dimensional arrays of dimension at least rn and n^2, respectively. NF and NP are the corresponding two-dimensional vectors giving the number of rows and columns in F and P. Any input requirements depend on and are specified by whether DISCREG, CNTNREG, or RICTNWT is employed by ASYMREG. When F data are input, it should be kept in mind that, because of the use of duality theory, ASYMREG will treat A' as A and H' as B. An initial stabilizing gain F for use in RICTNWT must cause (A' - H'F) to be asymptotically stable and be appropriately dimensioned.
IDENT	Logical variable: TRUE If G is an identity matrix FALSE Otherwise If G is an identity matrix, no data need be input for G, but G and NG must still appear as arguments of the calling sequence.
DISC	Logical variable: TRUE If the digital version is solved FALSE For the continuous version
NEWT	Logical variable: TRUE RICTNWT is used to solve the appropriate steady-state covariance equation FALSE Either CNTNREG or DISCREG is used depending upon the value of DISC.

STABLE Logical variable. Data for STABLE are not required if NEWT = FALSE, but STABLE must still appear as an argument of the calling sequence. When NEWT = TRUE:

STABLE = TRUE	If it is known that the matrix (A' - H'F) computed from input data is stable relative to an input parameter ALPHA.
STABLE = FALSE	(A' - H'F) is computed and tested (within ASYMREG) for stability relative to ALPHA.

FNULL Logical variable. Data for FNULL are not required if NEWT = FALSE, but FNULL must still appear as an argument of the calling sequence. When NEWT = TRUE:

FNULL = TRUE	If the input F is a null matrix
FNULL = FALSE	Otherwise

ALPHA SCALAR VARIABLE. ALPHA is not required if NEWT = FALSE or STABLE = TRUE. Otherwise, ALPHA is used in the asymptotic stability test of (A' - H'F) from input data.

IOP Five-dimensional option vector:

IOP(1) = 0	Do not print results.
Otherwise	Print input, GQG', F, P, the eigenvalues of P, the matrix (A - FH), and the eigenvalues of (A - FH).

IOP(2), IOP(3), IOP(4), and IOP(5) are the first four elements of the IOP vector of ASYMREG.

DUMMY Vector of working space for computations, dimensioned at least:

DISC	NEWT		
TRUE	TRUE	$6n^2 + 4n + 2$	(STABLE = FALSE)
		$5n^2$	(STABLE = TRUE)
TRUE	FALSE	$4n^2$	
FALSE	FALSE	$17n^2 + n(r + 18)$	
FALSE	TRUE	$8n^2 + n(r + 1)$	(BILIN)
		$5n^2 + n(r + 4) + 2$	(BARSTW)

Output arguments:

Q Upon normal return, Q is replaced by GQG'. The dimension of Q must be at least n^2.

F Upon normal return, F contains the filter gain for either the continuous or the discrete problem.

NF Two-dimensional vector giving, upon normal return, the number of rows and columns of F:

NF(1) = n
NF(2) = r

P	Upon normal return, P contains the steady-state covariance matrix.
NP	Two-dimensional vector giving the number of rows and columns of P. Upon normal return, NP = NA.
DUMMY	Upon normal return, the first n elements of DUMMY contain the eigenvalues of P, the next n^2 contain the matrix (A - FH) for filter gain F, and the next 2n contain the eigenvalues of (A - FH) stored as an $n \times 2$ matrix with the real parts as first column and imaginary parts as the second. All matrices are packed by columns into one-dimensional arrays.

COMMON blocks: None

Error messages: None directly from ASYMFIL

Field length: 713 octal words (459 decimal)

Subroutines employed by ASYMFIL: LNCNT, PRNT, TRANP, EQUATE, ASYMREG

Subroutines employing ASYMFIL: None

Comments: Cases in which G, with dimension $n \times m$, has $m \geq n$ can be treated as follows. Compute GQG' externally and, using subroutine FACTOR, find a $q \times n$ matrix D $(q \leq n)$ such that

$$GQG' = D'D$$

Then apply ASYMFIL with G replaced by D' and $Q = I_n$.

Extensions of the basic optimal filter problem considered in ASYMFIL (such as colored noise, singular R matrices, and correlated process and measurement noise) can be found in the literature (e.g., ref. 4). Generally, each extension can be solved by an appropriate combination of ASYMFIL with other subroutines of the ORACLS program. The transient Kalman-Bucy filter problem in which t_0 and i_0 are finite can be solved by using duality theory and the transient regulator solution capability of subroutines CNTNREG and DISCREG.

Subroutine EXPMDFL

Description: The purpose of EXPMDFL is to solve either the continuous or discrete time-invariant asymptotic explicit (model-in-the-system) model-following problem (ref. 28).

For the continuous case, the state and output equations are given as

$$\dot{x}(t) = A\ x(t) + B\ u(t)$$

$$y(t) = H\ x(t)$$

where $x(0) = x_0$ is given and the constant matrices A, B, and H are of dimension $n \times n$, $n \times r$ $(r \leq n)$, and $m \times n$ $(m \leq n)$, respectively. The control function $u(t)$ is required to minimize

$$J = \lim_{t_1 \to \infty} \left\{ \int_0^{t_1} \left[e'(t)\ Q\ e(t) + u(t')\ R\ u(t) \right] dt \right\}$$

where

$$e(t) = y(t) - y_m(t)$$

$$y_m(t) = H_m\ x_m(t)$$

and

$$\dot{x}_m(t) = A_m\ x_m(t)$$

where $x_m(0) = x_m^0$ is given. The constant matrices H_m and A_m have dimension $m \times \ell$ $(m \leq \ell)$ and $\ell \times \ell$, respectively. Also, $Q = Q' \geq 0$ and $R = R' > 0$. The optimization of the performance index causes the output $y(t)$ of the state to track the output $y_m(t)$ of a prescribed model. After substituting $e(t)$ into the performance index, the model-following problem can be transformed into choosing $u(t)$ to minimize

$$J = \lim_{t_1 \to \infty} \int_0^{t_1} \left[\tilde{x}'(t)\ \tilde{Q}\ \tilde{x}(t) + u'(t)\ R\ u(t) \right] dt$$

with

$$\dot{\tilde{x}}(t) = \tilde{A}\ \tilde{x}(t) + \tilde{B}\ u(t)$$

where

$$\tilde{A} = \begin{bmatrix} A & 0 \\ 0 & A_m \end{bmatrix}$$

$$\tilde{B} = \begin{bmatrix} B \\ 0 \end{bmatrix}$$

$$\tilde{Q} = \begin{bmatrix} H'QH & -H'QH_m \\ -H_m'QH & H_m'QH_m \end{bmatrix}$$

and

$$\tilde{x} = \begin{bmatrix} x \\ x_m \end{bmatrix}$$

This transformed problem can be solved directly using optimal linear regulator theory. If the $(\tilde{A},\tilde{B})$ pair is stabilizable and the $(\tilde{A},D)$ pair (with $D'D = \tilde{Q}$) is detectable, the solution exists and is given by

$$u(t) = -F\,\tilde{x}(t) = -F_{11}\,x(t) - F_{12}\,x_m(t)$$

Computationally, it is inefficient to work with the composite $(\tilde{A},\tilde{B})$ system directly. If the steady-state Riccati equation is formed and $\tilde{A}$, $\tilde{B}$, and $\tilde{Q}$ are substituted, it readily follows (ref. 28) that

$$F_{11} = R^{-1}B'P_{11}$$

with $P_{11} = P_{11}' \geq 0$ satisfying

$$P_{11}A + A'P_{11} - P_{11}BR^{-1}B'P_{11} + H'QH = 0$$

and

$$F_{12} = R^{-1}B'P_{12}$$

with P_{12} satisfying

$$P_{12}A_m + (A - BF_{11})'P_{12} = H'QH_m$$

The computation of (F_{11},F_{12}) thus separates into two parts:

(1) Evaluate the feedback gain F_{11} on the state x by solving a reduced-order optimal regulator problem of the form

$$\dot{x}(t) = A\ x(t) + B\ v(t)$$

$$y(t) = H\ x(t)$$

$$\min_{v(t)} \left\{ \lim_{t_1 \to \infty} \int_0^{t_1} \left[y'(t)\ Q\ y(t) + v'(t)\ R\ v'(t) \right] dt \right\}$$

leading to

$$v(t) = -F_{11}\ x(t)$$

(2) Using F_{11} from step (1), compute the feedforward gain F_{12} on the model x_m from the linear equations

$$P_{12}A_m + (A - BF_{11})'P_{12} = H'QH_m$$

$$RF_{12} = B'P_{12}$$

For the discrete case, the state and output equations are given as

$$x(i+1) = A\ x(i) + B\ u(i)$$

$$y(i) = H\ x(i)$$

with A, B, and H as previously defined. The control sequence $u(i)$ $(i = 0,1,\ldots,N-1)$ is required to minimize

$$J = \lim_{N\to\infty}\left\{\sum_{i=0}^{N-1}\left[e'(i+1)\ Q\ e(i+1) + u'(i)\ R\ u(i)\right]\right\}$$

where

$$e(i) = y(i) - y_m(i)$$

$$y_m(i) = H_m\ x_m(i)$$

$$x_m(i+1) = A_m\ x_m(i)$$

with Q, R, H_m, and A_m as previously defined. As in the continuous case, the discrete model-following problem can be solved in terms of a $(\tilde{A},\tilde{B},\tilde{Q},\tilde{R})$ optimal regulator formulation, but a simplified computational algorithm also exists (ref. 15):

(1) Compute a feedback gain F_{11} on the state x by solving the reduced-order optimal regulator problem

$$x(i+1) = A\ x(i) + B\ v(i)$$

$$y(i) = H\ x(i)$$

$$\min_{v(i)}\left\{\lim_{N\to\infty}\sum_{i=0}^{N-1}\left[y'(i+1)\ Q\ y(i+1) + v'(i)\ R\ v(i)\right]\right\}$$

leading to

$$v(i) = -F_{11}\ x(i)$$

(2) Using (P_{11},F_{11}) from step (1), compute a feedforward gain F_{12} on the model state x_m from the linear equations,

$$P_{12} = (A - BF_{11})'P_{12}A_m - H'QH_m$$

$$(B'P_{11}B + R)F_{12} = B'P_{12}A_m$$

The complete optimal model-following control law is then given by

$$u(i) = -F_{11}\, x(i) - F_{12}\, x_m(i)$$

EXPMDFL solves both the continuous and discrete versions of the explicit model-following problem through the simplified computational approach of solving a reduced-order regulator program for F_{11} and a set of linear equations for F_{12}. Subroutine ASYMREG is used to solve the reduced-order regulator problem, thus giving the user the choice of solving the appropriate steady-state Riccati equation for P_{11} by either of the subroutines DISCREG, CNTNREG, or RICTNWT. The subroutine BARSTW is used to solve the P_{12} equation in the continuous case and subroutine SUM in the discrete case. Final F_{12} computation in both cases employs subroutine SYMPDS. Computational parameters for BARSTW are set internally through the COMMON block TOL of subroutine RDTITL. An option is provided to bypass the (P_{11},F_{11}) computation and, using input (P_{11},F_{11}) data, proceed directly to the computation of (P_{12},F_{12}).

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL EXPMDFL(A,NA,B,NB,H,NH,AM,NAM,HM,NHM,Q,NQ,R,NR,F,NF, P,NP,HIDENT,HMIDENT,DISC,NEWT,STABLE,FNULL,ALPHA,IOP,DUMMY)

Input arguments:

A, B, H, AM, HM, Q, R	Matrices packed by columns in one-dimensional arrays; not destroyed upon normal return
NA, NB, NH, NAM, NHM, NQ, NR	Two-dimensional vectors giving the number of rows and columns of the respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon normal return
F	Matrix packed by columns into an array dimensioned at least $r(n + \ell)$. The first rn elements of F are treated as an input matrix F to ASYMREG for solving the reduced-order Riccati equation in P_{11}. If no input F data are required by ASYMREG, no data need be input for F in EXPMDFL unless the reduced-order Riccati computation for (P_{11},F_{11}) is to be bypassed. In this case, the first rn elements of F should contain a matrix F_{11} to be used in the computation of $(A - BF_{11})$.
NF	Two-dimensional vector giving, if required, the number of rows and columns of F used by ASYMREG or F_{11}: NF(1) = r NF(2) = n

P	Matrix packed by columns into an array dimensioned at least $n(n + \ell)$. The first n^2 elements of P are treated as an input matrix P to ASYMREG when used to solve the reduced-order Riccati equation for P_{11}. If no input P data are required by ASYMREG, no data need be input for P in EXPMDFL unless the reduced-order Riccati computation for (P_{11},F_{11}) is to be bypassed in the discrete case. In the discrete case, the first n^2 elements must contain a matrix to be used for P_{11} in the computation of the discrete F_{12}.
NP	Two-dimensional vector giving, if required, the number of rows and columns of P used by ASYMREG or P_{11}: NP(1) = n NP(2) = n
HIDENT	Logical variable: TRUE If H is an identity matrix FALSE Otherwise If H is an identity matrix, no data need be input for H, but H and NH must still appear as arguments of the calling sequence.
HMIDENT	Logical variable: TRUE If H_m (HM) is an identity matrix FALSE Otherwise If H_m is an identity matrix, no data need be input for HM, but HM and NHM must still appear as arguments of the calling sequence.
DISC	Logical variable: TRUE If the discrete version is solved FALSE For the continuous version
NEWT, STABLE, FNULL, ALPHA	Variables whose input values are determined by the choice of method to solve the steady-state Riccati equation for (P_{11},F_{11}) called for in ASYMREG; not required if the (P_{11},F_{11}) computation is bypassed but still must appear as arguments of the calling sequence
IOP	Five-dimensional option vector: IOP(1) = 0 Do not print results. Otherwise Print input and computed results. IOP(2) = 0 Do not compute (P_{11},F_{11}) but use input P and F as these variables. Otherwise Compute P_{11} and F_{11} through ASYMREG. IOP(3), IOP(4), and IOP(5) are first three elements of the IOP vector in ASYMREG; not required if IOP(2) = 0.

DUMMY Vector of working space for computations, dimensioned at least:

IOP(2)	DISC	
0	TRUE	$3n^2 + \ell^2 + \max(n^2,\ell n)$
0	FALSE	$3n^2 + 4n + 2 + \max(n^2,\ell n)$
Nonzero	TRUE or FALSE	Either n^2 plus the DUMMY requirement of ASYMREG or the preceding requirements for IOP(2) = 0, whichever is larger

Output arguments:

F	Upon normal return, $F = \left[F_{11},F_{12}\right]$ packed by columns in a one-dimensional array
NF	Upon normal return, NF(1) = r NF(2) = n + ℓ
P	For IOP(2) $\neq$ 0, $P = \left[P_{11},P_{12}\right]$ upon normal return. If IOP(2) = 0, the first $n\ell$ elements contain the matrix P_{12}. In both cases, P is packed by columns into a one-dimensional array.
NP	Upon normal return, NP(1) = n NP(2) = n + ℓ (IOP(2) $\neq$ 0) or NP(2) = ℓ (IOP(2) = 0)

COMMON block: TOL

Error message: If either R or $(B'P_{11}B+R)$ fails to be positive definite, the message "IN EXPMDFL, THE COEFFICIENT MATRIX FOR SYMPDS IS NOT SYMMETRIC POSITIVE DEFINITE" is printed, and the program is returned to the calling point.

Field length: 1516 octal words (846 decimal)

Subroutines employed by EXPMDFL: LNCNT, PRNT, EQUATE, ASYMREG, MULT, SUBT, TRANP, BARSTW, SUM, ADD, SYMPDS, SCALE

Subroutines employing EXPMDFL: None

Comments: If the model dynamics in both the continuous and discrete cases contain a direct link to the control, then

$$\tilde{B} = \begin{bmatrix} B \\ \\ B_m \end{bmatrix} \qquad (B_m \neq 0)$$

and the computation will not generally decouple into a simplified algorithm. In this case, ASYMREG can be applied to the composite $(\tilde{A},\tilde{B},\tilde{Q},R)$ system directly. For the transient case where N and t_1 are finite, the time-varying explicit model-following solution can be obtained using the composite $(\tilde{A},\tilde{B},\tilde{Q},R)$ system and subroutine DISCREG or CNTNREG.

Subroutine IMPMDFL

Description: The purpose of IMPMDFL is to solve either the continuous or discrete time-invariant asymptotic implicit (model-in-the-performance-index) model-following problem.

For the continuous case, the state and output equations are given as

$$\dot{x}(t) = A\,x(t) + B\,u(t)$$

$$y(t) = H\,x(t)$$

where $x(0) = x_0$ is given and the constant matrices A, B, and H are of dimension $n \times n$, $n \times r$ $(r \leq n)$, and $m \times n$ $(m \leq n)$, respectively. The control function $u(t)$ is required to minimize

$$J = \lim_{t_1 \to \infty} \left\{ \int_0^{t_1} \left[e'(t)\,Q\,e(t) + u'(t)\,R\,u(t) \right] dt \right\}$$

where

$$e(t) = \dot{y}(t) - A_m\,y(t) - B_m\,u(t)$$

The constant matrices A_m and B_m have dimension $m \times m$ and $m \times r$, respectively. Also, $Q = Q' \geq 0$ and $R = R' > 0$. The philosophy of implicit model following is to cause the output $y(t)$ of the system state to behave similar to a model state with dynamics

$$\dot{x}_m(t) = A_m\,x_m(t) + B_m\,u(t)$$

where $x_m(0) = \overset{o}{x}_m$ is given. This is done by forcing, in a weighted least squares sense, $y(t)$ to satisfy the model dynamic equation. Substituting for $e(t)$ in the performance index reduces the problem to choosing $u(t)$ to minimize

$$J = \lim_{t_1 \to \infty} \left\{ \int_0^{t_1} \left[x'(t)\,\tilde{Q}\,x(t) + x'(t)\,\tilde{W}\,u(t) + u'(t)\,\tilde{R}\,u(t) \right] dt \right\}$$

where

$$\tilde{Q} = (HA - A_mH)'Q(HA - A_mH)$$

$$\tilde{W} = 2(HA - A_mH)'Q(HB - B_m)$$

and

$$\tilde{R} = (HB - B_m)'Q(HB - B_m) + R$$

Applying the control transformation

$$u(t) = -F_0\, x(t) + v(t)$$

with

$$F_0 = \tilde{R}^{-1} \frac{\tilde{W}'}{2}$$

further reduces the problem to choosing $v(t)$ to minimize

$$J = \lim_{t_1 \to \infty} \left\{ \int_0^{t_1} \left[x'(t)\, \hat{Q}\, x(t) + v'(t)\, \tilde{R}\, v(t) \right] dt \right\}$$

where

$$\hat{Q} = \tilde{Q} - \frac{\tilde{W}}{2} F_0$$

$$\tilde{A} = A - BF_0$$

and

$$\dot{x}(t) = \tilde{A}\, x(t) + B\, v(t)$$

If the (A,B) pair is stabilizable and the $(\tilde{A},D)$ pair (with $D'D = \hat{Q}$) is detectable, a solution to the asymptotic implicit model-following problem exists and is given by

$$v(t) = -F_1\, x(t)$$

$$F_1 = \tilde{R}^{-1}B'P$$

$$P\tilde{A} + \tilde{A}'P + \hat{Q} - PB\tilde{R}^{-1}B'P = 0$$

and

$$u(t) = -F\, x(t)$$

with

$$F = F_0 + F_1$$

For the discrete case, the state and output equations are given as

$$x(i+1) = A\, x(i) + B\, u(i)$$

$$y(i) = H\, x(i)$$

where $x(0) = x_0$ is given and A, B, and H are as previously defined. The control function $u(i)$ $(i = 0,1,\ldots,N-1)$ is required to minimize

$$J = \lim_{N\to\infty} \left\{ \sum_{i=0}^{N-1} \left[e'(i)\, Q\, e(i) + u'(i)\, R\, u(i) \right] \right\}$$

where

$$e(i) = y(i+1) - A_m\, y(i) - B_m\, u(i)$$

The matrices A_m, B_m, Q, and R are as previously defined. Substituting for $e(i)$ in the performance index reduces the problem to choosing $u(i)$ to minimize

$$J = \lim_{N\to\infty} \left\{ \sum_{i=0}^{N-1} \left[x'(i)\, \tilde{Q}\, x(i) + x'(i)\, \tilde{W}\, u(i) + u'(i)\, \tilde{R}\, u(i) \right] \right\}$$

Applying the control transformation,

$$u(i) = -F_0\, x(i) + v(i)$$

further reduces the problem to choosing $v(i)$ to minimize

$$J = x'(0)\, \hat{Q}\, x(0) + \lim_{N\to\infty} \left\{ \sum_{i=0}^{N-1} \left[x'(i+1)\, \hat{Q}\, x(i+1) + v'(i)\, \tilde{R}\, v(i) \right] \right\}$$

with

$$x(i+1) = \tilde{A}\, x(i) + B\, v(i)$$

If the aforementioned stabilizability and detectability conditions are satisfied

$$v(i) = -F_1\, x(i)$$

$$F_1 = (\tilde{R} + B'PB)^{-1} B'P\, \tilde{A}$$

$$P = \phi' P \phi + F_1'\, \tilde{R}\, F_1 + \hat{Q}$$

$$\phi = \tilde{A} - BF_1$$

and

$$u(i) = -F\, x(i)$$

with

$$F = F_0 + F_1$$

The program IMPMDFL solves the steady-state Riccati equation to generate the appropriate P and F_1 through use of the subroutine ASYMREG, thus giving the user the choice of either subroutine DISCREG, CNTNREG, and RICTNWT to solve the final Riccati equation. An option is provided to bypass the (P,F_1) computation and return after computing $\tilde{Q}$, $\tilde{W}$, $\tilde{R}$, F_0, $\hat{Q}$, and $\tilde{A}$.

Source of software: Ernest S. Armstrong, LaRC

Calling sequence: CALL IMPMDFL(A,NA,B,NB,H,NH,AM,NAM,BM,NBM,Q,NQ,R,NR,F,NF,P,NP,IDENT,DISC,NEWT,STABLE,FNULL,ALPHA,IOP,DUMMY)

Input arguments:

A, B, H, AM, BM, Q, R	Matrices packed by columns in one-dimensional arrays; not destroyed upon normal return
NA, NB, NH, NAM, NBM, NQ, NR	Two-dimensional vectors giving the number of rows and columns of respective matrices; for example, NA(1) = Number of rows of A NA(2) = Number of columns of A Not destroyed upon normal return
F, NF, P, NP	F and P are matrices packed by columns into one-dimensional arrays of dimension at least rn and n^2, respectively. NF and NP are the corresponding two-dimensional vectors giving the number of rows and columns of F and P. Any input requirements depend on and are specified by whether DISCREG, CNTNREG, or RICTNWT is employed by ASYMREG. When F data are input for RICTNWT, it should be such that the input $(\tilde{A},B)$ system is stabilized.
IDENT	Logical variable: TRUE If H is an identity matrix FALSE Otherwise If H is an identity matrix, no data need be input for H, but H and NH must still appear as arguments of the calling sequence.
DISC	Logical variable: TRUE If the discrete version is solved FALSE For the continuous version
NEWT, STABLE, FNULL, ALPHA	Variables whose input values are determined by the choice of method called for from ASYMREG to solve the steady-state Riccati equation for P and F

IOP	Four-dimensional option vector: IOP(1) = 0 Do not print results. Otherwise Print input and computed results. IOP(2), IOP(3), and IOP(4) are the first three elements of the IOP vector in ASYMREG. If IOP(2) = -1000, the IMPMDFL program returns just prior to employing ASYMREG, in which case data for F, NF, P, NP, DISC, NEWT, STABLE, FNULL, ALPHA, IOP(3), and IOP(4) need not be entered, but these arguments must still appear in the calling sequence.
DUMMY	Vector of working space for computations, dimensioned at least $4n^2$ plus the dimension requirements for DUMMY in ASYMREG when (P,F_1) is to be computed. If the (P,F_1) computation is bypassed, DUMMY should be dimensioned at least $6n^2 + r$.

Output arguments:

F	If (P,F_1) is computed, $F = F_0 + F_1$ upon normal return. For (P,F_1) not computed, the input F is returned. All matrices are packed by columns into a one-dimensional array.
NF	Upon normal return, NF(1) = r NF(2) = n
P	If (P,F_1) is computed, P contains, upon normal return, the solution to the steady-state Riccati equation defining F_1. For (P,F_1) not computed, the input P is returned. All matrices are packed by columns into one-dimensional arrays.
NP	Upon normal return, NP(1) = n NP(2) = n when (P,F_1) is computed. Otherwise, the input NP is returned.
DUMMY	Upon normal return, if IOP(2) = -1000, the first n^2 elements contain the matrix $\hat{Q}$ and the next nr contain the matrix F_0. After $2n^2$ elements, the next r^2 contain the matrix $\tilde{R}$. After $3n^2$, the next n^2 contain the matrix $\tilde{A}$. Additionally, for (P,F_{11}) computed, after $4n^2$, the elements of DUMMY in IMPMDFL contain the elements of DUMMY returned by ASYMREG. All matrices are packed by columns as one-dimensional arrays.

COMMON blocks: None

Error messages: None directly from IMPMDFL

Field length: 1556 octal words (878 digital)

Subroutines employed by IMPMDFL: LNCNT, PRNT, SUBT, EQUATE, PREFIL, ASYMREG, ADD, MULT, TRANP, SCALE

Subroutines employing IMPMDFL: None

Comments: For the case in which t_1 or N is finite, set IOP(2) = -1000 and return with $\hat{Q}$, F_0, $\tilde{R}$, and $\tilde{A}$ in DUMMY. Then, using the $(\tilde{A},B,\hat{Q},\tilde{R})$ system, solve for the transient $P(\ )$ and $F_1(\ )$ from subroutine DISCREG or CNTNREG. The model-following control law is then

$$F(\ ) = F_0 + F_1(\ )$$

SUPPORTING SUBROUTINES

Subroutine READ1

Description: The purpose of READ1 is to input from cards and print or print without card input a single matrix A without employing a header card as in subroutine READ. When card input is employed, each row of the matrix starts on a new card using format (8F10.2). Printing is performed using subroutine PRNT.

Source of software: VASP (ref. 16) with modifications by Ernest S. Armstrong, LaRC

Calling sequence: CALL READ1(A,NA,NZ,NAM)

Input arguments:

A	Matrix packed by columns in a one-dimensional array; required as input only if NZ(1) = 0
NA, NZ	Two-dimensional arrays generally giving the number of rows and columns of A: NA(1) = NZ(1) = Number of rows of A NA(2) = NZ(2) = Number of columns of A Alternatively, if NZ(1) = 0, the data for (A,NA) are assumed to have been previously stored in locations A and NA. Otherwise, data for NA are not required as input.
NAM	Hollerith data: a four-character symbol used by PRNT

Output arguments:

A, NA	For NZ(1) ≠ 0, the matrix A which was read from cards and packed by columns into a one-dimensional array; upon normal return, NA(1) = NZ(1) NA(2) = NZ(2)

COMMON blocks: None

Error message: If either NZ(1) or NZ(1) × NZ(2) is less than unity, the message "ERROR IN READ1 MATRIX _____ HAS NA = _____, _____" is printed, and the program is returned to the calling point.

Field length: 133 octal words (91 decimal)

Subroutines employed by READ1: LNCNT, PRNT

Subroutine employing READ1: READ

Comments: None

Subroutine BALANC

Description: The purpose of BALANC is to balance a real square matrix for the calculation of eigenvalues and eigenvectors and isolate the eigenvalues whenever possible. The computational method follows that of Parlett and Reinsch found in reference 17. The subroutine BALANC is a translation of the ALGOL procedure BALANCE found on pages 315 to 326 of reference 2.

Source of software: LaRC Analysis and Computation Division subprogram library

Calling sequence: CALL BALANC(NM,N,A,LOW,IGH,SCALE)

Input arguments:

NM	First dimension of array A as given in the calling program
N	Number of rows of matrix A
A	Matrix which is to be balanced stored in a real two-dimensional array. The contents of this array are destroyed upon return.

Output arguments:

A	Upon normal return, A contains the balanced matrix.
LOW, IGH	Two integers such that A(I,J) = 0 if I > J and J = 1,2,...,LOW-1 or I = IGH+1,IGH+2,...,N.
SCALE	A one-dimensional array of dimension at least N. SCALE contains information determining the permutations and scaling factors used: suppose that the principal submatrix in rows LOW through IGH has been balanced, that P(J) denotes the index interchanged with J during the permutation step, and that the elements of the diagonal matrix used are denoted by D(I,J); then, SCALE(J) = P(J) (J = 1,2,...,LOW-1) = D(J,J) (J = LOW,LOW+1,...,IGH) = P(J) (J = IGH+1,IGH+2,...,N) The order in which the interchanges are made is N to IGH + 1, then 1 to LOW-1.

COMMON blocks: None

Error messages: None directly from BALANC

Field length: 327 octal words (215 decimal)

Subroutines employed by BALANC: None

Subroutine employing BALANC: EIGEN

Comments: None

Subroutine ELMHES

Description: Given a real square matrix A, the purpose of ELMHES is to reduce a submatrix situated in rows and columns LOW through IGH to upper Hessenberg form by stabilized elementary similarity transformations. The computational method follows that of Martin and Wilkinson in reference 18. ELMHES is a translation of the ALGOL procedure ELMHES found on pages 339 to 358 of reference 2.

Source of software: LaRC Analysis and Computation Division subprogram library

Calling sequence: CALL ELMHES(NM,N,LOW,IGH,A,INT)

Input arguments:

NM	First dimension of the array A as given in the calling program
N	Number of rows of matrix A
LOW, IGH	Integers typically determined by the balancing subroutine BALANC for eigenvalue and eigenvector computation. If BALANC is not used, set LOW = 1 and IGH = N.
A	Matrix which is to be used in the reduction to upper Hessenberg form stored in a real two-dimensional array. The contents of A are destroyed upon return.

Output arguments:

A	Upon normal return, A contains the upper Hessenberg matrix. The multipliers which were used in the reduction are stored in the remaining triangle under the Hessenberg matrix.
INT	One-dimensional integer array of dimension at least IGH in the calling program. INT contains information on the rows and columns interchanged in the reduction. Only elements LOW to IGH are used.

COMMON blocks: None

Error messages: None directly from ELMHES

Field length: 206 octal words (134 decimal)

Subroutines employed by ELMHES: None

Subroutine employing ELMHES: EIGEN

Comments: None

Subroutine HQR

Description: The purpose of HQR is to find the eigenvalues of a real square upper Hessenberg matrix H by the QR algorithm. The computational method follows that of Martin, Peters, and Wilkinson found in reference 19. The subroutine HQR is a translation of the ALGOL procedure HQR found on pages 359 to 371 of reference 2.

Source of software: LaRC Analysis and Computation Division subprogram library

Calling sequence: CALL HQR(NM,N,LOW,IGH,H,WR,WI,IERR)

Input arguments:

NM	First dimension of array H as given in the calling program
N	Number of rows in matrix H
LOW, IGH	Integers typically determined by the balancing subroutine BALANC. If BALANC is not used, set LOW = 1 and IGH = N.
H	Matrix in upper Hessenberg form stored in a real two-dimensional array. Information about the transformations used in the reduction to Hessenberg form by subroutine ELMHES, if performed, is stored in the remaining triangle under the Hessenberg matrix. Upon normal return, H is destroyed.

Output arguments:

WR, WI	One-dimensional arrays, dimensioned at least N in the calling program, containing, upon normal return, the real and imaginary parts of the eigenvalues, respectively. The eigenvalues are unordered except that the complex conjugate pairs of values appear consecutively with the eigenvalue having positive imaginary part first.
IERR	Integer error code: IERR = 0 Normal return IERR = J The Jth eigenvalue has not been determined after 30 iterations of the QR algorithm. If an error exit is made, the eigenvalues should be correct for indices IERR+1,IERR+2,...,N.

COMMON blocks: None

Error messages: None directly from HQR; IERR should be examined upon return.

Field length: 556 octal words (366 decimal)

Subroutines employed by HQR: None

Subroutine employing HQR: EIGEN

Comments: None

Subroutine INVIT

Description: The purpose of INVIT is to find those eigenvectors of a real square upper Hessenberg matrix corresponding to specified eigenvalues using inverse iteration. The subroutine INVIT is a translation of the ALGOL procedure INVIT found on pages 418 to 439 of reference 2.

Source of software: LaRC Analysis and Computation Division subprogram library

Calling sequence: `CALL INVIT(NM,N,A,WR,WI,SELECT,MM,M,Z,IERR,RM1,RV1,RV2)`

Input arguments:

NM	First dimension of the array A as given in the calling program
N	Number of rows in A
A	Matrix in upper Hessenberg form stored in real two-dimensional array. Upon return, A is unaltered.
WR, WI	One-dimensional arrays dimensioned at least N by the calling program. WR and WI contain the real and imaginary parts, respectively, of the eigenvalues of the matrix. The eigenvalues must be stored in a manner identical to that of subroutine HQR. WI is unaltered upon return. WR may be altered since close eigenvalues are perturbed slightly in searching for independent eigenvectors.
SELECT	One-dimensional array of logical variables, dimensioned at least N by the calling program. SELECT specifies the eigenvectors to be found. The eigenvector corresponding to the Jth eigenvalue is specified by setting SELECT(J) to TRUE. Upon return, SELECT may be altered. If the elements corresponding to a pair of conjugate complex eigenvalues were each initially set TRUE, the program resets the second of the two elements to FALSE.
MM	MM should be set to an upper bound for the number of columns required to store the eigenvectors to be found. Note that two columns are required to store the eigenvector corresponding to a complex eigenvalue.
RM1, RV1, RV2	Temporary storage arrays dimensioned at least N × N, N, and N, respectively, by the calling program

Output arguments:

M	The actual number of columns used to store the eigenvectors

Z Matrix dimensioned at least NM × MM by the calling program. Z contains the real and imaginary parts of the eigenvectors. If the next selected eigenvalue is real, the next column of Z contains its eigenvector. If the eigenvalue is complex, the next two columns of Z contain the real and imaginary parts of its eigenvector. The eigenvectors are normalized so that the component of largest magnitude is unity. Any vector which fails the acceptance test is set to zero.

IERR Integer error code:

IERR = 0	Normal return
IERR = -(2N + 1)	More than MM columns of Z are necessary to store the eigenvectors corresponding to the specified eigenvalues.
IEhR = -K	The iteration corresponding to the Kth value fails.
IERR = -(N + K)	Both of the above error situations occur.

COMMON blocks: None

Error messages: None directly from INVIT; IERR should be examined upon return.

Field length: 1310 octal words (712 decimal)

Subroutines employed by INVIT: None

Subroutine employing INVIT: EIGEN

Comments: None

Subroutine ELMBAK

Description: The purpose of ELMBAK is to form the eigenvectors of a real square matrix A by back transforming those of the corresponding upper Hessenberg matrix determined by subroutine ELMHES. ELMBAK is a translation of the ALGOL procedure ELMBAK found on pages 339 to 358 of reference 2.

Source of software: LaRC Analysis and Computation Division subprogram library

Calling sequence: CALL ELMBAK(NM,LOW,IGH,A,INT,M,Z)

Input arguments:

NM	First dimension of the array A as given in the calling program
LOW, IGH	Integers available from the balancing subroutine BALANC. If BALANC is not used, set LOW = 1 and IGH = The order of the matrix.
A	Two-dimensional array dimensioned at least NM × IGH in the calling program. A contains the multipliers which were used in the reduction by ELMHES in its lower triangle below the subdiagonal.
INT	One-dimensional array dimensioned at least IGH by the calling program. INT contains information on the rows and columns interchanged in the reduction by ELMHES. Only elements LOW through IGH are used.
M	The number of columns of Z to be back transformed
Z	Two-dimensional array dimensioned at least NM × M by the calling program. Z contains the real and imaginary parts of the eigenvectors to be back transformed in its first M columns. The contents of Z are destroyed upon return.

Output argument:

Z	The real and imaginary parts of the transformed eigenvectors are returned in the first M columns.

COMMON blocks: None

Error messages: None directly from ELMBAK

Field length: 133 octal words (91 decimal)

Subroutines employed by ELMBAK: None

Subroutine employing ELMBAK: EIGEN

Comments: None

Subroutine BALBAK

Description: The purpose of BALBAK is to form the eigenvectors of a real square matrix by back transforming those of the corresponding balanced matrix determined by subroutine BALANC. BALBAK is a translation of the ALGOL procedure BALBAK found on pages 315 to 326 of reference 2.

Source of software: LaRC Analysis and Computation Division subprogram library

Calling sequence: CALL BALBAK(NM,N,LOW,IGH,SCALE,M,Z)

Input arguments:

NM	First dimension of the matrix array as given in the calling program
N	Number of rows of the matrix
LOW, IGH	Integers determined by BALANC
SCALE	One-dimensional array dimensioned at least N by the calling program. SCALE contains information determining the permutations and scaling factors used by BALANC.
M	The number of columns of Z to be back transformed
Z	Two-dimensional array dimensioned at least NM × M in the calling program. Z contains the real and imaginary parts of the eigenvectors to be back transformed in its first M columns.

Output argument:

Z	The real and imaginary parts of the transformed eigenvectors are returned in the first M columns.

COMMON blocks: None

Error messages: None directly from BALBAK

Field length: 124 octal words (84 decimal)

Subroutines employed by BALBAK: None

Subroutine employing BALBAK: EIGEN

Comments: None

Subroutine DETFAC

Description: The purpose of DETFAC is to factor a real square matrix A as

$$PA = LU$$

where P is a permutation matrix representing row pivotal strategy, L is a unit lower triangular matrix, and U is an upper triangular matrix. Options are provided to compute the determinant of A with and without A input in factored form.

Source of software: LaRC Analysis and Computation Division subprogram library with modifications by Ernest S. Armstrong, LaRC

Calling sequence: CALL DETFAC(NMAX,N,A,IPIVOT,IDET,DETERM,ISCALE,WK,IERR)

Input arguments:

NMAX	First dimension of the array A as given in the calling program
N	Number of rows of matrix A
A	Two-dimensional array dimensioned at least NMAX × N in the calling program. A is the matrix to be factored. If the factored form of A is input, $A = (L\backslash U)$ should be used neglecting the unity elements of L and the pivotal strategy input through the array IPIVOT. For A in unfactored form, input data are destroyed.
IPIVOT	One-dimensional array dimensioned at least N by the calling program. Not required as input if the unfactored form of A is used. Otherwise, IPIVOT(I) = J indicates that row J of matrix A was used to pivot for the Ith column.
IDET	Determinant evaluation code: 0 Compute L and U matrices only. 1 Given L and U matrices as input, compute parameters defining the determinant. 2 Compute L, U, and determinant parameters.
WK	One-dimensional array dimensioned at least N by the calling program and used as a work storage array.

Output arguments:

A	Upon normal return, the L and U matrices are over stored in A as $A = (L\backslash U)$ neglecting the unity elements in L.
IPIVOT	Upon normal return, IPIVOT contains the pivotal strategy as previously explained.
DETERM, ISCALE	Determinant evaluation parameters; upon return, for $IDET \neq 0$, $\det(A) = DETERM \times 10^{100 \times ISCALE}$
IERR	Singularity test parameter: 0 Matrix A is singular. 1 Matrix A is nonsingular.

COMMON blocks: None

Error messages: None directly from DETFAC; IERR should be examined upon return.

Field length: 332 octal words (218 decimal)

Subroutines employed by DETFAC: None

Subroutine employing DETFAC: GELIM

Comments: None

Subroutine AXPXB

Description: The purpose of AXPXB is to solve the real matrix equation

$$AX + XB = C$$

where A, B, and C are constant matrices of order $m \times m$, $n \times n$, and $m \times n$, respectively. The matrices A and B are transformed into real lower and upper Schur form (ref. 5), and the transformed system is solved by back substitution. The option is provided to input the Schur forms directly and bypass the Schur decomposition.

Source of software: Coded directly from reference 5

Calling sequence: `CALL AXPXB(A,U,M,NA,NU,B,V,N,NB,NV,C,NC,EPSA,EPSB,FAIL)`

Input arguments:

A	Two-dimensional array dimensioned at least $(m + 1) \times (m + 1)$ in the calling program. The upper $m \times m$ part contains the matrix A. If the Schur form of A is input directly the lower triangle and superdiagonal of the upper $m \times m$ part of the array A contain a lower Schur form of A.
U	Two-dimensional array dimensioned at least $m \times m$ in the calling program. Not required as input if the Schur form of A is not input directly. Otherwise, U contains the orthogonal matrix that reduces A to Schur form $\tilde{A}$ via $\tilde{A} = U'AU$
M	Number of rows of the matrix A
NA	First dimension of the array A; at least $m + 1$
NU	First dimension of the array U; at least m
B	Two-dimensional array dimensioned at least $(n + 1) \times (n + 1)$ in the calling program. The upper $n \times n$ part contains the matrix B. If the Schur form of B is input directly, the upper triangle and subdiagonal of the upper $n \times n$ part of the array B contain an upper real Schur form of B.
V	Two-dimensional array dimensioned at least $n \times n$ in the calling program. Not required as input if the Schur form of A is not input directly. Otherwise, V contains the orthogonal matrix that reduces B to Schur form $\tilde{B}$ via $\tilde{B} = V'BV$
N	Number of rows of the matrix B

NB	First dimension of the array B; at least $n + 1$
NV	First dimension of the array V; at least n
C	Two-dimensional array dimensioned at least $m \times n$ by the calling program; destroyed upon return
NC	First dimension of the array C; at least m
EPSA	Convergence criterion for the reduction of A to Schur form. EPSA should be set slightly smaller than 10^{-k_a} where k_a is the number of significant digits in the elements of A. Set EPSA negative if a Schur form for A and transformation matrix U are directly input.
EPSB	Convergence criterion for the reduction of B to Schur form. EPSB should be set slightly smaller than 10^{-k_b} where k_b is the number of significant digits in the elements of B. Set EPSB negative if a Schur form for B and transformation matrix V are directly input.

Output arguments:

A	Upon normal return, the array A contains the lower Schur form of the matrix A.
U	Upon normal return, the array U contains the orthogonal matrix U which transforms A to Schur form.
B	Upon normal return, the array B contains the upper Schur form for the matrix B.
V	Upon normal return, the array V contains the orthogonal matrix V which transforms B to upper Schur form.
C	Upon normal return, the solution X is stored in C.
FAIL	Integer variable containing an error signal. If FAIL is positive (negative) then the program was unable to reduce A(B) to real Schur form. If FAIL = 0, the reductions proceeded without mishap.

COMMON blocks: None

Error messages: None directly from AXPXB; FAIL should be tested upon return.

Field length: 667 octal words (439 decimal)

Subroutines employed by AXPXB: HSHLDR, BCKMLT, SCHUR, SHRSLV

Subroutine employing AXPXB: BARSTW

Comments: Subroutine AXPXB should be used when $AX + XB = C$ needs to be solved for a number of C matrices. For the first C, set EPSA and EPSB based on k_a and k_b. Afterward, assuming FAIL = 0, set EPSA and EPSB to negative values and compute X for the remaining C matrices. For the special case in which

$$A = B' \quad \text{and} \quad C = C'$$

the subroutine ATXPXA should be employed.

Subroutine SHRSLV

Description: The purpose of SHRSLV is to solve the real matrix equation

$$AX + XB = C$$

where A is an $m \times m$ matrix in lower real Schur form and B is an $n \times n$ matrix in upper real Schur form.

Source of software: Coded directly from reference 5

Calling sequence: `CALL SHRSLV(A,B,C,M,N,NA,NB,NC)`

Input arguments:

A	Two-dimensional array dimensioned at least $m \times m$ in the calling program. A contains the lower Schur form of the matrix A. Not destroyed upon return.
B	Two-dimensional array dimensioned at least $n \times n$ in the calling program. B contains the matrix B in upper Schur form. Not destroyed upon return.
C	Two-dimensional array dimensioned at least $m \times n$ by the calling program. C contains the C matrix of the algebraic equation which is destroyed upon return.
M	Number of rows of the matrix A
N	Number of rows of the matrix B
NA	First dimension of the array A; at least m
NB	First dimension of the array B; at least n
NC	First dimension of the array C; at least m

Output argument:

C	Upon normal return, the solution X is in C.

COMMON block: SLVBLK

Error messages: None directly from SHRSLV

Field length: 444 octal words (292 decimal)

Subroutine employed by SHRSLV: SYSSLV

Subroutine employing SHRSLV: AXPXB

Comments: None

Subroutine ATXPXA

Description: The purpose of ATXPXA is to solve the real matrix equation

$$A'X + XA = C$$

where A and C are constant matrices of dimension $n \times n$ with

$$C = C'$$

The matrix A is transformed into upper Schur form (ref. 5) and the transformed system is solved by back substitution. The option is provided to input the Schur form directly and bypass the Schur decomposition.

Source of software: Coded directly from reference 5

Calling sequence: CALL ATXPXA(A,U,C,N,NA,NU,NC,EPS,FAIL)

Input arguments:

A	Two-dimensional array dimensioned at least $(n + 1) \times (n + 1)$ in the calling program. The upper $n \times n$ part contains the matrix A. If the Schur form of A is input directly the upper triangle and the first subdiagonal of the upper $n \times n$ part of the array A contain an upper real Schur form of A.
U	Two-dimensional array dimensioned at least $n \times n$ in the calling program. Not required as input if the Schur form of A is not input directly. Otherwise, U contains the orthogonal matrix that reduces A to Schur form $\tilde{A}$ via $\tilde{A} = U'AU$
C	Two-dimensional array dimensioned at least $n \times n$ by the calling program; destroyed upon return
N	Number of rows of the matrix A
NA	First dimension of the array A; at least $n + 1$
NU	First dimension of the array U; at least n
NC	First dimension of the array C; at least n
EPS	Convergence criterion for the reduction of A to Schur form. EPS should be set slightly smaller than 10^{-k_a} where k_a is the number of significant digits in the elements of A. Set EPS negative if a Schur form for A and transformation matrix U are directly input.

Output arguments:

A	Upon normal return, the array A contains the upper Schur form of the matrix A. Same as input if EPS < 0.
U	Upon normal return, the array U contains the orthogonal matrix U which transforms A to Schur form. Same as input if EPS < 0.
C	Upon normal return, the solution X is stored in C.
FAIL	Integer variable containing an error signal. If FAIL is nonzero, the program was unable to reduce A to real Schur form. If FAIL = 0, the reduction proceeded without mishap.

COMMON blocks: None

Error messages: None directly from ATXPXA. FAIL should be tested upon return.

Field length: 562 octal words (370 decimal)

Subroutines employed by ATXPXA: HSHLDR, BCKMLT, SCHUR, SYMSLV

Subroutine employing ATXPXA: BARSTW

Comments: Subroutine ATXPXA should be used when

$$A'X + XA = C$$

needs to be solved for a number of symmetric C matrices. For the first C, set EPS based on k_a. Afterward, assuming FAIL = 0, set EPS negative and compute X for the remaining C matrices.

Subroutine SYMSLV

Description: The purpose of SYMSLV is to solve the real matrix equation

$$A'X + XA = C$$

where $C = C'$ and A is $n \times n$ and in upper real Schur form.

Source of software: Coded directly from reference 5

Calling sequence: CALL SYMSLV(A,C,N,NA,NC)

Input arguments:

A	Two-dimensional array dimensioned at least $n \times n$ in the calling program. A contains the upper Schur form for the matrix A. Not destroyed upon return.
C	Two-dimensional array dimensioned at least $n \times n$ in the calling program. C contains the C matrix of the algebraic equation which is destroyed upon return.
N	Number of rows of the matrix A
NA	First dimension of the array A; at least n
NC	First dimension of the array C; at least n

Output argument:

C	Upon normal return, the solution X is stored in C.

COMMON block: SLVBLK

Error messages: None directly from SYMSLV

Field length: 535 octal words (349 decimal)

Subroutine employed by SYMSLV: SYSSLV

Subroutine employing SYMSLV: ATXPXA

Comments: None

Subroutine HSHLDR

Description: The purpose of HSHLDR is to reduce a real n × n matrix A to upper Hessenberg form by Householder's method of elementary Hermitian transformations described in reference 29.

Source of software: Coded directly from reference 5

Calling sequence: CALL HSHLDR(A,N,NA)

Input arguments:

A	Two-dimensional array dimensioned at least (n + 1) × (n + 1) in the calling program. The upper n × n part of the array A contains the matrix A which is destroyed upon return.
N	Number of rows of matrix A
NA	First dimension of the array A; at least n + 1

Output argument:

A	Upon normal return, the upper triangle of the array A to column n contains the upper triangle of the Hessenberg form of the matrix A. Column n + 1 contains the subdiagonal elements of the Hessenberg form. The lower triangle and the (n + 1)th row of the array contain a history of the Householder transformations.

COMMON blocks: None

Error messages: None directly from HSHLDR

Field length: 265 octal words (181 decimal)

Subroutines employed by HSHLDR: None

Subroutines employing HSHLDR: AXPXB, ATXPXA

Comments: None

Subroutine BCKMLT

Description: Given the output A from subroutine HSHLDR, the purpose of BCKMLT is to compute the orthogonal matrix that reduces A to upper Hessenberg form.

Source of software: Coded directly from reference 5

Calling sequence: CALL BCKMLT(A,U,N,NA,NU)

Input arguments:

A	Two-dimensional array containing the output from HSHLDR
N	Number of rows in matrix A in HSHLDR
NA	First dimension of the array A; at least N + 1
NU	First dimension of the array U; at least N

Output argument:

U	Two-dimensional array dimensioned at least n × n where n is the order of the A matrix in HSHLDR. If the matrix A is used for U in the calling sequence, the elements of the orthogonal matrix will overwrite the output of HSHLDR.

COMMON blocks: None

Error messages: None directly from BCKMLT

Field length: 201 octal words (129 decimal)

Subroutines employed by BCKMLT: None

Subroutines employing BCKMLT: AXPXB, ATXPXA

Comments: None

Subroutine SCHUR

Description: The purpose of SCHUR is to reduce an $n \times n$ upper Hessenberg matrix H to real Schur form. Computation is by the QR algorithm with implicit origin shifts. The program SCHUR is an adaptation of the ALGOL program HQR found in reference 19. The product of the transformations used in the reduction is accumulated.

Source of software: Coded directly from reference 5

Calling sequence: CALL SCHUR(H,U,NN,NH,NU,EPS,FAIL)

Input arguments:

H	Two-dimensional array dimensioned at least $n \times n$ in the calling program. The array H contains the matrix H in upper Hessenberg form. The elements below the third subdiagonal are undisturbed upon return.
U	Two-dimensional array dimensioned at least $n \times n$ by the calling program. On input, U contains any square matrix desired.
NN	The number of rows in the matrices H and U
NH	First dimension of the array H; at least n
NU	First dimension of the array U; at least n
EPS	Number used in determining when an element of H is negligible. The element H(i,j) is negligible if $\lvert H(i,j) \rvert \le EPS \times \lVert H \rVert$ where $\lVert H \rVert$ denotes the ℓ_∞ norm of H.

Output arguments:

H	Upon normal return, H contains an upper Schur form of H.
U	Upon normal return, U contains the product of the input U right multiplied by the accumulated orthogonal transformations used to reduce H to Schur form. If the identity matrix is input for U, then the Schur form is U'HU
FAIL	Integer variable containing an error signal. If FAIL is positive, then the program failed to make the (FAIL-1) or (FAIL-2) subdiagonal element negligible after 30 iterations of the QR algorithm.

COMMON blocks: None

Error messages: None directly from SCHUR

Field length: 520 octal words (336 decimal)

Subroutines employed by SCHUR: None

Subroutines employing SCHUR: AXPXB, ATXPXA

Comments: None

Subroutine SYSSLV

Description: The purpose of SYSSLV is to solve the linear system

$$Ax = b$$

where A is an $n \times n$ ($n \leq 5$) matrix and b is n-dimensional vector. Solution is by Crout reduction. The matrix A, the vector b, and order n are contained in the arrays A, B, and the variable N of the COMMON block SLVBLK. The solution is returned in the array B.

Source of software: Coded directly from reference 5

Calling sequence: CALL SYSSLV

Input arguments: None

Output arguments: None

COMMON block: SLVBLK

Error messages: None directly from SYSSLV

Field length: 227 octal words (151 decimal)

Subroutines employed by SYSSLV: None

Subroutines employing SYSSLV: SHRSLV, SYMSLV

Comments: None

Subroutine GAUSEL

Description: The purpose of GAUSEL is to solve a set of linear equations,

$$AX = B$$

by the method of Gaussian elimination. The constant matrices A and B are of dimension $n \times n$ and $n \times r$, respectively. No information is returned on the pivotal strategy or the value of the determinant of A.

Source of software: LaRC Analysis and Computation Division subprogram library

Calling sequence: `CALL GAUSEL(MAX,N,A,NR,B,IERR)`

Input arguments:

MAX	First dimension of the array B given in the calling program
N	Number of rows of the matrix A
A	Two-dimensional array dimensioned at least $n \times n$ in the calling program; destroyed upon return
NR	Number of columns in the matrix B
B	Two-dimensional array dimensioned at least MAX × NR in the calling program; destroyed upon return

Output arguments:

B	Upon normal return, B contains the solution X.
IERR	Integer error code: 0 Normal return 2 Input matrix A is singular.

COMMON blocks: None

Error messages: None directly from GAUSEL. IERR should be checked upon return.

Field length: 317 octal words (207 decimal)

Subroutines employed by GAUSEL: None

Subroutine employing GAUSEL: EXPADE

Comments: None

EXAMPLE COMPUTATIONS

This section contains selected examples illustrating the use of the ORACLS program subroutines. Sample executive programs and output data are presented which develop state variable feedback control laws using the optimal transient regulator, optimal sampled-data regulator, and model-following design approaches. Additionally, the construction of an asymptotic Kalman-Bucy estimator is illustrated.

Data employed for the plant equations are from a linearized mathematical model of the lateral dynamics of an F-8 aircraft, presented in reference 30. The problem and machine-dependent accuracy and convergence parameters required for COMMON blocks TOL and CONV of subroutine RDTITL used in the example computations are:

EPSAM = EPSBM = 1.E-10

IACM = 12

SUMCV = 1.E-8

MAXSUM = 50

RICTCV = 1.E-8

SERCV = 1.E-8

The construction of RDTITL for the example computations is shown in figure 1. All computations were performed using the Control Data Cyber digital computer system under Network Operating System (NOS) 1.2.

```
      SUBROUTINE RDTITL
C
      COMMON/LINES/NLP,LIN,TITLE(8),TIL(2)
      COMMON/FORM/NEPR,FMT1(6),FMT2(6)
      COMMON/TOL/EPSAM,EPSBM,IACM
      COMMON/CONV/SUMCV,MAXSUM,RICTCV,SERCV
C NPL = NO. LINES/PAGE VARIES WITH THE INSTALLATION
      DATA LIN,NLP/1,44/
      DATA NEPR,FMT1/7,10H(1P7E16.7)/
      DATA TIL/10H   ORACL,10HS   PROGRAM/
      DATA FMT2/10H(3X,1P7E16,10H.7)       /
      DATA EPSAM/1.E-10/
      DATA EPSBM/1.E-10/
      DATA IACM/12/
      DATA SUMCV/1.E-8/
      DATA RICTCV/1.E-8/
      DATA SERCV/1.E-8/
      DATA MAXSUM/50/
      READ(5,100) TITLE
      IF(EOF(5))90,91
   90 CONTINUE
      STOP 1
   91 CONTINUE
  100 FORMAT(8A10)
      CALL LNCNT(100)
      RETURN
      END
```

Figure 1.- Subroutine RDTITL for the example computations.

Example 1 - Optimal Transient Regulator

This problem illustrates the construction of storage arrays for ORACLS and demonstrates the solution of a simple transient optimal linear regulator problem. Given the system

$$\dot{x}(t) = A\ x(t) + B\ u(t)$$

with

$$A = \begin{bmatrix} -2.6 & 0.25 & -38. & 0.0 \\ -0.075 & -0.27 & 4.4 & 0.0 \\ 0.078 & -0.99 & -0.23 & 0.052 \\ 1.0 & 0.078 & 0.0 & 0.0 \end{bmatrix}$$

and

$$B = \begin{bmatrix} 17. & 7.0 \\ 0.82 & -3.2 \\ 0.0 & 0.046 \\ 0.0 & 0.0 \end{bmatrix}$$

find the control law $u(t) = -F(t)\ x(t)$ which minimizes

$$J = \frac{1}{2}\ x'(20)\ x(20) + \int_0^{20} \left[x'(t)\ x(t) + 100\ u'(t)\ u(t)\right]\ dt$$

The solution can be found by directly employing the subroutine CNTNREG. Output data are to be printed at 2.0-second intervals between 0 and 20.

Referring to the description of CNTNREG, the dimensions of the subroutine argument arrays must be specified. The matrices A, B, Q, and R are 4 × 4, 4 × 2, 4 × 4, and 2 × 2, respectively. As packed one-dimensional arrays, they must be dimensioned at least A(16), B(8), Q(16), and R(4). However, for this example, input data to CNTNREG are defined within the source program and, for convenience of construction, two-dimensional arrays A(4,4), B(4,2), Q(4,4), and R(2,2) are used. Since $H = I_4$, no data for H are required if the logical variable IDENT is set to TRUE. The array P initially contains the weighting matrix $\frac{1}{2} I_4$ on the final states and hence, is dimensioned P(16). The array F will contain the feedback gain matrix and must be dimensioned at least F(8). Similarly, Z, W, LAMBDA, and S are dimensioned 64, 64, 16, and 16, respectively. Dimensions for NA, NB, NQ, NR, NF, and NP are set to 2. The vector T contains the final problem time and print interval and is also dimensioned 2. The vector IOP controls the printing from CNTNREG and must be dimensioned at least 3. The dimension of DUMMY is computed from the formula (since IOP(3) = 0 for a transient solution to the Riccati equation),

$$9n^2 + 17n + nr$$

with $n = 4$ and $r = 2$.

The following executive program shows how the specific data for CNTNREG may be constructed. Note that subroutines NULL and UNITY may be employed to generate zero and unity elements in the coefficient and weighting matrices. For the NOS 1.2 system at LaRC tape 5 is designated input and tape 6 output, as indicated by the source program card. Output from ORACLS follows the executive program.

```
      PROGRAM  REGLAT(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C
      DIMENSION A(4,4),B(4,2),Q(4,4),R(2,2),Z(64),W(64),LAMBDA(16),S(16)
     1,F(2,4),P(16),DUMMY(220),NA(2),NB(2),NQ(2),NR(2),IOP(3),T(2),NF(2)
     2,NP(2)
      LOGICAL IDENT
      REAL LAMBDA
C
C
C     INITIALIZE NA,NB,....,NP
      NA(1)= 4
      NA(2)= 4
      NB(1)= 4
      NB(2)= 2
      NQ(1)= 4
      NQ(2)= 4
      NR(1)= 2
      NR(2)= 2
      NP(1)= 4
      NP(2)= 4
C
C     SET FINAL TIME AND PRINT INTERVAL
      T(1)=20.
      T(2) = 2.
C
C     DEFINE PRINT AND TRANSIENT SOLUTION OPTIONS
      IOP(1) = 1
      IOP(2) = 1
      IOP(3) = 0
C
C     DEFINE COEFFICIENT AND WEIGHTING MATRICES
      CALL NULL(A,NA)
      CALL NULL(B,NB)
      CALL UNITY(P,NP)
      CALL SCALE(P,NP,P,NP,0.5)
      CALL UNITY(Q,NQ)
      CALL UNITY(R,NR)
      CALL SCALE(R,NR,R,NR,100.)
C
      A(1,1)= -2.6
      A(1,2) = .25
      A(1,3)=-38.
      A(2,1) =  -.075
      A(2,2) =  -.27
      A(2,3) =  4.4
      A(3,1) =  .076
      A(3,2) =  -.99
      A(3,3) =  -.23
      A(3,4) =  .052
      A(4,1) =  1.0
      A(4,2) =  .078
      B(1,1) =  17.
      B(1,2) =  7.0
      B(2,1) =  .82
      B(2,2) =  -3.2
      B(3,2) =  .046
      IDENT = .TRUE.
C
C     INPUT HOLLERITH DATA FOR TITLE OF OUTPUT
      CALL RDTITL
C
C     NOW USE CNTNREG TO SOLVE THE TRANSIENT REGULATOR PROBLEM
      CALL CNTNREG(A,NA,B,NB,H,NH,C,NQ,R,NR,Z,W,LAMBDA,S,F,NF,P,NP,T,IOP
     1,IDENT,DUMMY)
C
C
      STOP
      END
     EXAMPLE 1 - OPTIMAL TRANSIENT REGULATOR                                              ORACLS  PROGRAM


PROGRAM TO SOLVE THE TIME-INVARIANT FINITE-DURATION CONTINUOUS OPTIMAL
REGULATOR PROBLEM WITH NOISE-FREE MEASUREMENTS

      A     MATRIX         4 ROWS        4 COLUMNS
 -2.6000000E+00   2.5000000E-01  -3.8000000E+01   0.
 -7.5000000E-02  -2.7000000E-01   4.4000000E+00   0.
  7.8000000E-02  -9.9000000E-01  -2.3000000E-01   5.2000000E-02
  1.0000000E+00   7.8000000E-02   0.              0.

      B     MATRIX         4 ROWS        2 COLUMNS
  1.7000000E+01   7.0000000E+00
  8.2000000E-01  -3.2000000E+00
  0.              4.6000000E-02
  0.              0.

      Q     MATRIX         4 ROWS        4 COLUMNS
  1.0000000E+00   0.              0.              0.
  0.              1.0000000E+00   0.              0.
  0.              0.              1.0000000E+00   0.
  0.              0.              0.              1.0000000E+00

H IS AN IDENTITY MATRIX


      R     MATRIX         2 ROWS        2 COLUMNS
  1.0000000E+02   0.
  0.              1.0000000E+02


WEIGHTING ON TERMINAL VALUE OF STATE VECTOR


      P     MATRIX         4 ROWS        4 COLUMNS
  5.0000000E-01   0.              0.              0.
  0.              5.0000000E-01   0.              0.
  0.              0.              5.0000000E-01   0.
  0.              0.              0.              5.0000000E-01

      Z     MATRIX         8 ROWS        8 COLUMNS
     EXAMPLE 1 - OPTIMAL TRANSIENT REGULATOR                                              ORACLS  PROGRAM

  -2.6000000E+00   2.5000000E-01  -3.8000000E+01   0.              -3.3800000E+00   8.4600000E-02  -3.2200000E-03
     0.
  -7.5000000E-02  -2.7000000E-01   4.4000000E+00   0.               9.4600000E-02  -1.0912400E-01   1.4720000E-03
     0.
   7.8000000E-02  -9.9000000E-01  -2.3000000E-01   5.2000000E-02  -3.2200000E-03   1.4720000E-03  -2.1160000E-05
     0.
   1.0000000E+00   7.9000000E-02   0.              0.              0.              0.              0.
     0.
  -1.0000000E+00   0.              0.              0.               2.6000000E+00   7.5000000E-02  -7.8000000E-02
    -1.0000000E+00
   0.             -1.0000000E+00   0.              0.              -2.5000000E-01   2.7000000E-01   9.9000000E-01
    -7.8000000E-02
   0.              0.             -1.0000000E+00   0.               3.8000000E+01  -4.4000000E+00   2.3000000E-01
     0.
   0.              0.              0.             -1.0000000E+00   0.              0.             -5.2000000E-02
     0.


 EIGENVALUES OF Z


  -6.4605654E-01   0.
   6.4605654E-01   0.
   7.8466995E-01   2.6501143E+00
   7.8466995E-01  -2.6501143E+00
  -7.8466995E-01   2.6501143E+00
  -7.8466995E-01  -2.6501143E+00
  -2.8451457E+00   0.
   2.8451457E+00   0.


 CORRESPONDING EIGENVECTORS


  -2.4733621E-01  -1.7233188E-01   5.0257395E-02   2.6562832E-02  -8.5529327E-02  -1.6423059E-01   6.2134586E-01
    -6.2618216E-02
   4.3326566E-03  -3.3791761E-02  -1.6344167E-02   9.5183892E-03   3.5983643E-02   1.2843065E-02   3.0404894E-02
    -6.9930617E-03
   7.5286043E-03   7.7409397E-03   1.7516136E-06  -8.0834088E-03  -4.9824656E-03   1.7745854E-02  -2.6988189E-03
     3.3935252E-04
   3.8230750E-01  -2.7062403E-01   1.4511446E-02  -1.4186503E-02  -4.8131275E-02   4.9465498E-02  -2.1922162E-01
    -2.2200567E-02
```

146

```
    EXAMPLE 1 - OPTIMAL TRANSIENT REGULATOR                                                    ORACLS   PROGRAM

  7.2558584E-02   9.5507170E-02  -2.2377640E-02   2.4483836E-02  -1.5780566E-02  -4.4315091E-02   8.4269310E-02
    1.0441627E-01
  5.3890433E-01   7.9262267E-01   3.3965417E-01  -7.8633601E-03   3.2328081E-01  -7.4606776E-03   2.3749420E-01
    3.4982534E-01
 -4.3207089E-01   3.2205187E-01   1.7435213E-01   9.2129547E-01  -2.9506394E-01   8.7411594E-01  -7.0239213E-01
    9.2653245E-01
  5.5697883E-01   3.4327414E-01  -1.4120636E-02   4.7159165E-03  -3.8062689E-02  -1.2681699E-02  -8.9888548E-02
   -9.1676080E-03


REORDERED EIGENVECTORS


 -1.7233188E-01   5.3257395E-02   2.6512832E-02  -6.2619218E-02  -7.4733021E-01  -1.6423055E-01  -8.5528327E-02
    8.2134588E-01
 -3.3791761E-02  -1.5344167E-02   4.5183892E-03  -6.9930617E-03   4.3376566E-03   1.2843065E-02   3.5983643E-02
    3.6404904E-02
  7.7409397E-03   1.7515130E-05  -9.0834098E-03   3.3935252E-04   7.5786043E-03   1.7745854E-02  -4.9824656E-03
   -2.6988149E-03
 -2.7082403E-01   1.4511446E-02  -1.4186539E-02  -2.2200507E-02   3.8730750E-01   4.5465498E-02  -4.8131275E-02
   -2.1922182E-01
  9.5507170E-02  -2.2377640E-02   2.4483836E-02   1.0441627E-01   7.2558584E-02  -4.4315091E-02  -1.5780566E-02
    8.4269310E-02
  7.9262267E-01   3.3965417E-01  -7.8633601E-03   3.4982534E-01   5.3890433E-01  -7.4606776E-03   3.2328081E-01
    2.3749420E-01
  3.2205187E-01   1.7435213E-01   9.2129547E-01   9.2653245E-01  -4.3207089E-01   8.7411594E-01  -2.9506394E-01
   -7.0239213E-01
  3.4327414E-01  -1.4120636E-02   4.7159165E-03  -9.1676080E-03   5.5697883E-01  -1.2681699E-02  -3.8062689E-02
   -8.9888548E-02


LAMBDA MATRIX OF EIGENVALUES OF Z WITH POSITIVE REAL PARTS


  6.4605654E-01   0.              0.              0.
  0.              7.8466995E-01   2.6501143E+00   0.
  0.             -2.6501143E+00   7.8466995E-01   0.
  0.              0.              0.              2.8451457E+00

    WIZW  MATRIX       8 ROWS       8 COLUMNS
```

```
     EXAMPLE 1 - OPTIMAL TRANSIENT REGULATOR                                                    ORACLS   PROGRAM

  6.4605654E-01   4.3539694E-13  -1.6913328E-13  -2.0390208E-13  -7.3247951E-14  -3.4396140E-13   8.6190507E-14
     1.7480680E-13
  9.9622886E-14   7.8465995E-01   2.6501143E+00  -4.8159991E-13  -8.2746769E-13   4.7260138E-13   1.5520196E-12
     1.1224239E-13
  1.5597072E-13  -2.6501143E+00   7.8466995E-01   2.6422617E-12  -1.7036632E-13   1.1402066E-12  -1.4882895E-12
    -1.9369661E-12
 -6.5446563E-14  -9.1129449E-13  -7.4776612E-13   2.8451457E+00  -1.3339999E-13   4.7238794E-14   6.1187571E-13
    -1.2551633E-12
  6.0327566E-14  -1.2656352E-12  -3.2722815E-13   1.4561670E-12  -6.4605654E-01   8.1668150E-13  -4.3938717E-13
    -1.4901340E-12
  1.3295405E-13  -3.3147102E-12  -1.4740753E-13   2.4210317E-12  -3.4414850E-13  -7.8466995E-01  -2.6501143E+00
    -3.4247465E-12
 -2.2830483E-13   1.3227621E-12   3.0763466E-12  -1.7386355E-12   1.0132695E-12   2.6501143E+00  -7.8466995E-01
     9.5261412E-13
  1.6940885E-13  -1.7646861E-12  -5.3523563E-13   2.5110453E-12  -2.5555875E-13   7.2909408E-13  -1.8733266E-12
    -2.8451457E+00

     W11    MATRIX          4 ROWS          4 COLUMNS
 -1.7233188E-01   5.0257395E-02   2.6542832E-02  -6.2618216E-02
 -3.3791761E-02  -1.6344167E-02   9.5183842E-03  -6.9930617E-03
  7.7409397E-03   1.7516130E-06  -9.0834699E-03   3.3535292E-04
 -2.7082403E-01   1.4511446E-02  -1.4166539E-02  -2.2260507E-02

     W21    MATRIX          4 ROWS          4 COLUMNS
  9.5507170E-02  -2.2377640E-02   2.4483698E-02   1.0441627E-01
  7.9262267E-01   3.3465417E-01  -7.8633603E-03   3.4925394E-01
  3.2205167E-01   1.7435213E-01   9.2129547E-01   9.2953245E-01
  3.9327414E-01  -1.4123838E-02   4.7159165E-03  -4.1676068E-03

     W12    MATRIX          4 ROWS          4 COLUMNS
 -2.4733021E-01  -1.9423095E-01  -9.5528327E-02   6.2134588E-01
  4.3326566E-03   1.2843665E-02   3.5953643E-02   3.0404804E-02
  7.5266643E-03   1.7745954E-02  -4.9824652E-03  -2.6909169E-03
  3.8230756E-01   4.5463498E-02  -4.4131275E-02  -2.1922162E-01

     W22    MATRIX          4 ROWS          4 COLUMNS
  7.2558584E-02  -4.4315691E-02  -1.5780596E-02   8.4269310E-02
  5.3890433E-01  -7.4600776E-03   3.2328081E-01   2.3749420E-01
 -4.3207089E-01   6.7411594E-01  -2.9560894E-01  -7.0239213E-01
  5.5697883E-01  -1.2681699E-02  -3.8362694E-02  -8.9888540E-02

      S     MATRIX          4 ROWS          4 COLUMNS
     EXAMPLE 1 - OPTIMAL TRANSIENT REGULATOR

 -6.9197483E-01   8.8040174E-02  -7.0617792E-03  -1.0579571E-01
  5.3014743E-01   1.1451662E-01  -4.2235758E-01  -1.6582063E+00
  1.0292420E+00  -6.8625132E-01   8.0290849E-01  -1.3835163E-01
 -4.1882228E-01  -2.9967598E-01  -4.0340142E-01   1.2407188E+00

MATRIX (R INVERSE)X(B TRANSPOSE)


  1.7000000E-01   8.2000000E-03   0.              0.
  7.0000000E-02  -3.2000000E-02   4.6000000E-04   0.

EXP(-LAMBDA X    .20000000E+01)


  2.7468973E-01   0.              0.              0.
  0.              1.1545667E-01   1.7323718E-01   0.
  0.             -1.7323718E-01   1.1545067E-01   0.
  0.              0.              0.              3.3786088E-03


TIME =    .20000000E+02


     P    MATRIX        4 ROWS        4 COLUMNS
  5.0000000E-01   0.              0.              0.
  0.              5.0000000E-01   0.              0.
  0.              0.              5.0000000E-01   0.
  0.              0.              0.              5.0000000E-01


TIME =    .18000000E+02


     P    MATRIX        4 ROWS        4 COLUMNS
  2.2480611E-01   3.6054056E-01  -1.4217206E+00   3.2023881E-01
  3.6054050E-01   1.0744607E+01  -6.8655022E+00   1.5409135E+00
 -1.4217206E+00  -6.8655022E+00   4.5683890E+01  -2.6235118E+00
  3.2023881E-01   1.5409135E+00  -2.6235118E+00   1.5641109E+00
```

```
     EXAMPLE 1 - OPTIMAL TRANSIENT REGULATOR                                                    ORACLS  PROGRAM


     F     MATRIX          2 ROWS         4 COLUMNS
  4.1173471E-02   1.4939767E-01  -3.1438962E-01   6.7076089E-02
  3.5451406E-03  -3.2266773E-01   2.0519022E-01  -2.8099331E-02


TIME =     .16000000E+02


     P     MATRIX          4 ROWS         4 COLUMNS
  2.3819322E-01   4.0985673E-01  -1.5275098E+00   3.6636522E-01
  4.0985673E-01   1.0991594E+01  -9.0079421E+00   1.7136851E+00
 -1.5275098E+00  -9.0079421E+00   4.9256328E+01  -2.9811826E+00
  3.6636522E-01   1.7136851E+00  -2.9811826E+00   1.7228142E+00

     F     MATRIX          2 ROWS         4 COLUMNS
  4.3853673E-02   1.5980672E-01  -3.3354180E-01   7.6334306E-02
  2.8594559E-03  -3.2719471E-01   2.0398637E-01  -3.0563703E-02


TIME =     .14000000E+02


     P     MATRIX          4 ROWS         4 COLUMNS
  2.3924330E-01   4.1302539E-01  -1.5297598E+00   3.6980853E-01
  4.1302539E-01   1.1028965E+01  -9.0076508E+00   1.7278448E+00
 -1.5297598E+00  -9.0076508E+00   4.9271952E+01  -2.9877059E+00
  3.6980853E-01   1.7278448E+00  -2.9877059E+00   1.7346173E+00

     F     MATRIX          2 ROWS         4 COLUMNS
  4.4058169E-02   1.6065183E-01  -3.3392190E-01   7.7035778E-02
  2.8265289E-03  -3.2815862E-01   2.0382674E-01  -3.0778783E-02


TIME =     .12000000E+02


     P     MATRIX          4 ROWS         4 COLUMNS
  2.3930520E-01   4.1337855E-01  -1.5297546E+00   3.7003289E-01
```

```
     EXAMPLE 1 - OPTIMAL TRANSIENT REGULATOR                                                          ORACLS   PROGRAM

  4.1337855E-01   1.1032206E+01  -9.0096823E+00   1.7293204E+00
 -1.5297546E+00  -9.0096823E+00   4.9275911E+01  -2.9880218E+00
  3.7003289E-01   1.7293204E+00  -2.9880218E+00   1.7354617E+00

      F     MATRIX          2 ROWS          4 COLUMNS
  4.4071588E-02   1.6073844E-01  -3.3393766E-01   7.7086018E-02
  2.8195631E-03  -3.2623855E-01   2.0389393E-01  -3.0810439E-02


TIME =     .10000000E+02


      P     MATRIX          4 ROWS          4 COLUMNS
  2.3930917E-01   4.1341056E-01  -1.5297727E+00   3.7004881E-01
  4.1341056E-01   1.1032469E+01  -9.0098261E+00   1.7294493E+00
 -1.5297727E+00  -9.0098261E+00   4.9276275E+01  -2.9880973E+00
  3.7004881E-01   1.7294493E+00  -2.9880973E+00   1.7355256E+00

      F     MATRIX          2 ROWS          4 COLUMNS
  4.4072525E-02   1.6074604E-01  -3.3394193E-01   7.7089781E-02
  2.8188087E-03  -3.2624478E-01   2.0389743E-01  -3.0813485E-02


TIME =     .80000000E+01


      P     MATRIX          4 ROWS          4 COLUMNS
  2.3930948E-01   4.1341291E-01  -1.5297749E+00   3.7005005E-01
  4.1341291E-01   1.1032489E+01  -9.0098419E+00   1.7294590E+00
 -1.5297749E+00  -9.0098419E+00   4.9276293E+01  -2.9881061E+00
  3.7005005E-01   1.7294590E+00  -2.9881061E+00   1.7355306E+00

      F     MATRIX          2 ROWS          4 COLUMNS
  4.4072598E-02   1.6074661E-01  -3.3394244E-01   7.7090072E-02
  2.8187540E-03  -3.2824526E-01   2.0389779E-01  -3.0813714E-02


TIME =     .60000000E+01
```

```
     EXAMPLE 1 - OPTIMAL TRANSIENT REGULATOR                                      ORACLS  PROGRAM

     P     MATRIX         4 ROWS         4 COLUMNS
   2.3930951E-01   4.1341309E-01  -1.5297751E+00   3.7005015E-01
   4.1341309E-01   1.1032491E+01  -9.0098433E+00   1.7294597E+00
  -1.5297751E+00  -9.0098433E+00   4.9276294E+01  -2.9881068E+00
   3.7005015E-01   1.7294597E+00  -2.9881068E+00   1.7355310E+00

     F     MATRIX         2 ROWS         4 COLUMNS
   4.4072604E-02   1.6074665E-01  -3.3394248E-01   7.7090094E-02
   2.8187502E-03  -3.2824531E-01   2.0389783E-01  -3.0813729E-02


TIME =     .40000000E+01


     P     MATRIX         4 ROWS         4 COLUMNS
   2.3930951E-01   4.1341310E-01  -1.5297751E+00   3.7005015E-01
   4.1341310E-01   1.1032491E+01  -9.0098434E+00   1.7294597E+00
  -1.5297751E+00  -9.0098434E+00   4.9276294E+01  -2.9881068E+00
   3.7005015E-01   1.7294597E+00  -2.9881068E+00   1.7355310E+00

     F     MATRIX         2 ROWS         4 COLUMNS
   4.4072604E-02   1.6074665E-01  -3.3394248E-01   7.7090096E-02
   2.8187499E-03  -3.2824531E-01   2.0389783E-01  -3.0813730E-02

STEADY-STATE SOLUTION HAS BEEN REACHED IN CNTNREG
```

Example 2 - Optimal Sampled-Data Regulator

This problem demonstrates a situation in which several ORACLS subroutines are required to obtain a solution and input data are entered through use of the READ subroutine.

Given the linear time-invariant system

$$\dot{x}(t) = A\,x(t) + B\,u(t)$$

with $x(0) = x_0$ given and A and B as defined in example 1, find the control law

$$u(t) = -F\,x(t)$$

which minimizes

$$J = \lim_{t_1 \to \infty} \left\{ \int_0^{t_1} \left[x'(t)\,Q\,x(t) + u'(t)\,R\,u(t) \right] dt \right\}$$

where Q and R are 4 × 4 and 2 × 2 identity matrices, respectively, and u(t) is restricted to be piecewise constant over uniform time intervals of measure $\Delta = 0.5$ second. Under these restrictions, the dynamical equations and performance index J become

$$x\left[(i+1)\Delta\right] = \tilde{A}\,x(i\Delta) + \tilde{B}\,u(i\Delta)$$

and

$$J = \lim_{N \to \infty} \left\{ \sum_{i=0}^{N} \left[x'(i\Delta)\,\tilde{Q}\,x(i\Delta) + x'(i\Delta)\,\tilde{W}\,u(i\Delta) + u'(i\Delta)\,\tilde{R}\,u(i\Delta) \right] \right\}$$

where

$$\tilde{A} = e^{A\Delta}$$

$$\tilde{B} = \int_0^{\Delta} e^{A\tau}\,B\,d\tau$$

$$\tilde{Q} = \int_0^{\Delta} e^{A'\tau}\; Q\; e^{A\tau}\; d\tau$$

$$\tilde{W} = 2 \int_0^{\Delta} e^{A'\tau}\; Q\; H(\tau,0)\; d\tau$$

$$\tilde{R} = \int_0^{\Delta} \Big[R + H'(\tau,0)\; Q\; H(\tau,0)\Big]\; d\tau$$

and

$$H(t,0) = \int_0^{t} e^{A\tau}\; B\; d\tau$$

Performing the control variable transformation,

$$u(i\Delta) = -\tilde{F}\; x(i\Delta) + v(i\Delta)$$

with

$$\tilde{F} = R^{-1}\, \frac{\tilde{W}'}{2}$$

leads to

$$x\Big[(i+1)\Delta\Big] = \hat{A}\; x(i\Delta) + \hat{B}\; v(i\Delta)$$

$$J = \lim_{N\to\infty} \left\{ \sum_{i=0}^{N} \Big[x'(i\Delta)\; \hat{Q}\; x(i\Delta) + v'(i\Delta)\; \hat{R}\; v(i\Delta)\Big] \right\}$$

where

$$\hat{A} = \tilde{A} - \tilde{B}\tilde{F}$$

$$\hat{B} = \tilde{B}$$

$$\hat{Q} = \tilde{Q} - \frac{\widetilde{WF}}{2}$$

$$\hat{R} = \tilde{R}$$

Ignoring the initial value $x(0)$ in J gives the problem of minimizing

$$\lim_{N \to \infty} \left(\sum_{i=0}^{N} \left\{ x'\left[(i+1)\Delta\right] \hat{Q}\, x\left[(i+1)\Delta\right] + v'(i\Delta)\, \hat{R}\, v(i\Delta) \right\} \right)$$

subject to

$$x\left[(i+1)\Delta\right] = \hat{A}\, x(i\Delta) + \hat{B}\, v(i\Delta)$$

whose solution, if it exists, is given by

$$v(i\Delta) = -\hat{F}\, x(i\Delta)$$

The gain matrix for the original problem is then

$$F = \tilde{F} + \hat{F}$$

The complete problem can be solved by appropriately combining the subroutines EXPINT (for $\hat{A}$ and $\hat{B}$), SAMPL (for $\tilde{Q}$, $\tilde{W}$, and $\tilde{R}$), PREFIL (for $\hat{A}$, $\hat{Q}$, and $\tilde{F}$), and ASYMREG (for $\hat{F}$). An executive program for constructing the solution follows. Data are input by using subroutine READ rather than being defined internally to the source program, as in example 1. The dimension of DUMMY is chosen to be the maximum of the requirements of each operation employing this storage vector. At the stage where matrices F and $\hat{F}$ are printed, the statement "CALL LNCNT (100)" is used to shift the printing to the top of the next page. Output data are presented after the executive program.

```
      PROGRAM SAMDAT(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C
      DIMENSION A(16),B(8),Q(16),R(4),ATIL(15),BTIL(8),QTIL(16),WTIL(8),
     1RTIL(4),FTIL(8),AHAT(16),QHAT(16),FHAT(8),DUMMY(116),P(16),F(8)
      DIMENSION NA(2),NB(2),NQ(2),NR(2),NATIL(2),NBTIL(2),NQTIL(2),NWTIL
     1(2),NRTIL(2),NFTIL(2),NAHAT(2),NQHAT(2),NFHAT(2),IOP(5),NP(2),NF(2
     2),NDUM(2)
C
      LOGICAL IDENT,DISC,NEWT,STABLE,FNULL
C
C
C     INPUT HOLLERITH DATA FOR TITLE OF OUTPUT
      CALL RDTITL
C
C     INPUT COEFFICIENT AND WEIGHTING MATRICES FOR CONTINUOUS SYSTEM
      CALL READ(4,A,NA,B,NB,Q,NQ,R,NR)
C
C
C     GENERATE EXP (A*DELT) AND (INTEGRAL EXP(A*TAU))*B
      DELT=.05
      IOP(1)=0
      N1 = (NA(1)**2)+1
C
      CALL EXPINT(A,NA,ATIL,NATIL,DUMMY,NDUM,DELT,IOP,DUMMY(N1))
      CALL MULT(DUMMY,NDUM,B,NB,BTIL,NBTIL)
      CALL PRNT(ATIL,NATIL,4HATIL,1)
      CALL PRNT(BTIL,NBTIL,4HBTIL,1)
C
C
C     GENERATE DIGITAL PERFORMANCE INDEX WEIGHTING MATRICES
      IOP(2)=1
      CALL EQUATE(Q,NQ,QTIL,NQTIL)
      CALL EQUATE(R,NR,RTIL,NRTIL)
      CALL SAMPL(A,NA,B,NB,QTIL,NQTIL,RTIL,NRTIL,WTIL,NWTIL,DELT,IOP,DUM
     1MY)
      CALL PRNT(QTIL,NQTIL,4HQTIL,1)
      CALL PRNT(WTIL,NWTIL,4HWTIL,1)
      CALL PRNT(RTIL,NRTIL,4HRTIL,1)
C
C
C     FIND PREFILTER GAIN WHICH ELEMINATES CROSS-PRODUCT TERM
C     IN DIGITAL PERFORMANCE INDEX
```

```
      IOP(3)=1
      CALL EQUATE(ATIL,NATIL,AHAT,NAHAT)
      CALL EQUATE(QTIL,NQTIL,QHAT,NQHAT)
      CALL PREFIL(AHAT,NAHAT,BTIL,NBTIL,QHAT,NQHAT,WTIL,NWTIL,RTIL,NRTIL
     1,FTIL,NFTIL,IOP,DUMMY)
      CALL PRNT(AHAT,NAHAT,4HAHAT,1)
      CALL PRNT(QHAT,NQHAT,4HQHAT,1)
      CALL PRNT(FTIL,NFTIL,4HFTIL,1)
C
C
C     SOLVE FOR F HAT
      IDENT= .TRUE.
      DISC = .TRUE.
      NEWT = .TRUE.
      STABLE = .FALSE.
      FNULL = .TRUE.
      ALPHA = .9
      IOP(1) = 1
      IOP(2) = 0
      IOP(3) = 0
      IOP(4) = 1
      IOP(5) = 1
      CALL ASYPREG(AHAT,NAHAT,BTIL,NBTIL,H,NH,QHAT,NQHAT,RTIL,NRTIL,FHAT
     1,NFHAT,P,NP,IDENT,DISC,NEWT,STABLE,FNULL,ALPHA,IOP,DUMMY)
      CALL ADD(FTIL,NFTIL,FHAT,NFHAT,F,NF)
C
C
C     OUTPUT F AND F HAT GAINS
      CALL LNCNT(100)
      PRINT 100
  100 FORMAT(*       FOR THE ORIGINAL SAMPLED-DATA PROBLEM*)
      CALL PRNT(FHAT,NFHAT,4HFHAT,1)
      CALL PRNT(F,NF,4HF   ,1)
C
C
      STOP
      END
     EXAMPLE 2 - OPTIMAL SAMPLED-DATA REGULATOR                                         ORACLS  PROGRAM

        A     MATRIX          4 ROWS          4 COLUMNS
  -2.6000000E+00   2.5000000E-01  -3.8000000E+01   0.
  -7.5000000E-02  -2.7000000E-01   4.4000000E+00   0.
   7.8000000E-02  -9.4000000E-01  -2.3000000E-01   5.2000000E-02
   1.0000000E+00   7.8000000E-02   0.              0.

        B     MATRIX          4 ROWS          2 COLUMNS
   1.7000000E+01   7.0000000E+00
   8.2000000E-01  -3.2000000E+00
   0.              4.6000000E-02
   0.              0.

        Q     MATRIX          4 ROWS          4 COLUMNS
   1.0000000E+00   0.              0.              0.
   0.              1.0000000E+00   0.              0.
   0.              0.              1.0000000E+00   0.
   0.              0.              0.              1.0000000E+00

        R     MATRIX          2 ROWS          2 COLUMNS
   1.0000000E+03   0.
   0.              1.0000000E+00

      ATIL   MATRIX          4 ROWS          4 COLUMNS
   8.7460216E-01   5.6206553E-02  -1.7644035E+00  -2.3524288E-03
  -3.76988872E-03  9.8114696E-01   2.1998505E-01   2.8616468E-04
         431137E-03 -4.8707923E-02   9.7950437E-01   2.5772852E-03
         07596E-02   4.9177112E-03  -4.4809768E-02   9.9996068E-01

      BTIL   MATRIX          4 ROWS          2 COLUMNS
   7.96926907E-01  3.2234606E-01
   3.9250218E-02  -1.5895733E-01
   6.1800654E-04   6.8676137E-03
   2.0435169E-02   7.9852045E-03

      QTIL   MATRIX          4 ROWS          4 COLUMNS
   4.3959404E-02   8.7398740E-04  -4.1390797E-02   1.1608794E-03
   8.7398740E-04   4.9227561E-02   2.8619327E-03   1.1266907E-04
  -4.1390797E-02   2.8619327E-03   1.0386723E-01  -6.3835981E-04
   1.1608794E-03   1.1266907E-04  -6.3835981E-04   4.9999178E-02

      WTIL   MATRIX          4 ROWS          2 COLUMNS
```

```
 3.7319029E-02   1.5229661E-02
 3.1985357E-03  -7.3838678E-03
-4.8263666E-02  -2.0667855E-02
 6.4086387E-04   2.4991144E-04

   RTIL  MATRIX        2 ROWS        2 COLUMNS
 6.0964766E-02   4.3451993E-03
 4.3451993E-03   5.2234106E-02

   AHAT  MATRIX        4 ROWS        4 COLUMNS
 5.9854577E-01   5.4764994E-02  -1.4049026E+00  -7.0633127E-03
 4.4954715E-03   9.6826103E-01   2.0868737E-01   3.9798041E-04
 2.7592406E-03  -4.8223984E-02   9.8096098E-01   2.5606174E-03
 3.9775917E-02   4.8601665E-03  -3.5637777E-02   9.9984043E-01

   QHAT  MATRIX        4 ROWS        4 COLUMNS
 3.7487559E-02   8.4516580E-04  -3.2962128E-02   1.0504445E-03
 8.4516580E-04   4.9906642E-02   2.8635932E-03   1.1174823E-04
-3.2962128E-02   2.8635932E-03   9.2886064E-02  -4.9458179E-04
 1.0504445E-03   1.1174823E-04  -4.9458179E-04   4.9997293E-02

   FTIL  MATRIX        2 ROWS        4 COLUMNS
 2.9744352E-01   3.1456861E-02  -3.8400852E-01   5.1158474E-03
 1.2103929E-01  -7.3297516E-02  -1.6589418E-01   1.9666527E-03


COMPUTATION OF F SUCH THAT A-BF IS ASYMPTOTICALLY STABLE IN THE DISCRETE SENSE


    A    MATRIX        4 ROWS        4 COLUMNS
 6.6505086E-01   6.0849943E-02  -1.5610029E+00  -7.8481252E-03
 4.9949683E-03   1.0758456E+00   2.3187486E-01   4.4220046E-04
 3.0659229E-03· -5.3582205E-02   1.0899566E+00   2.6451304E-03
 4.4195464E-02   5.4002072E-03  -3.9597530E-02   1.1109338E+00

    B    MATRIX        4 ROWS        2 COLUMNS
 8.8547102E-01   3.5816229E-01
 4.3611353E-02  -1.7661925E-01
 6.9667393E-04   7.6306819E-03
 2.2705743E-02   8.8725383E-03
```

```
ALPHA =     .33873839E+00


      F     MATRIX         2 ROWS         4 COLUMNS
  8.1498121E-01   2.2921930E+00  -3.0628590E+01   1.4899924E+01
  1.0955671E-01  -5.7422787E+00   8.0373720E+01  -3.4265557E+00

     A-BF  MATRIX         4 ROWS         4 COLUMNS
 -9.5830473E-02   9.7847250E-02  -3.2271096E+00  -1.1974036E+01
 -1.1197641E-02  -3.9317030E-02   1.5763176E+01  -1.2545593E+00
  1.6702042E-03  -1.1338692E-02   4.9766221E-01   1.8760698E-02
  2.4718663E-02   4.3028501E-03  -5.7271536E-02   8.0302221E-01


EIGENVALUES OF A-BF

     EIGN  MATRIX         4 ROWS         2 COLUMNS
  2.3419229E-01   3.3528007E-01
  2.3419229E-01  -3.3528007E-01
  3.4908617E-01   3.1664598E-01
  3.4908617E-01  -3.1664598E-01


MODULI OF EIGENVALUES OF A-BF


     MOD   MATRIX         4 ROWS         1 COLUMNS
  4.0897281E-01
  4.0897281E-01
  4.7130228E-01
  4.7130228E-01


PROGRAM TO SOLVE DISCRETE STEADY-STATE RICCATI EQUATION BY THE NEWTON ALGORITHM


      A     MATRIX         4 ROWS         4 COLUMNS
  5.9854577E-01   5.4764994E-02  -1.4049026E+00  -7.0633127E-03
  4.4954715E-03   9.6826103E-01   2.0868737E-01   3.9798041E-04
  2.7592406E-03  -4.8223984E-02   9.8096098E-01   2.5606174E-03
```

```
     EXAMPLE 2 - OPTIMAL SAMPLED-DATA REGULATOR

  3.9775917E-02   4.8601665E-03  -3.5637777E-02   9.9984043E-01

     B     MATRIX         4 ROWS        2 COLUMNS
  7.9692392E-01   3.2234606E-01
  3.9250218E-02  -1.5995733E-01
  6.1800654E-04   6.8676137E-03
  2.0435169E-02   7.9852845E-03

     Q     MATRIX         4 ROWS        4 COLUMNS
  3.7487559E-02   8.4516580E-04  -3.2962128E-02   1.0504445E-03
  8.4516580E-04   4.8906642E-02   2.8635932E-03   1.1174823E-04
 -3.2962128E-02   2.8635932E-03   9.2886064E-02  -4.9458179E-04
  1.0504445E-03   1.1174823E-04  -4.9458179E-04   4.9997293E-02

H IS AN IDENTITY MATRIX


     R     MATRIX         2 ROWS        2 COLUMNS
  6.0964766E-02   4.3451993E-03
  4.3451993E-03   5.2234106E-02

INITIAL F MATRIX


     F     MATRIX         2 ROWS        4 COLUMNS
  8.1498121E-01   2.2921930E+00  -3.0628590E+01   1.4899924E+01
  3.0955671E-01  -5.7422787E+00   8.0373720E+01  -3.4265557E+00


FINAL VALUES OF P AND F AFTER   10 ITERATIONS TO CONVERGE


     P     MATRIX         4 ROWS        4 COLUMNS
  5.1520273E-02   1.0590966E-02  -9.5089721E-02   5.6173730E-02
  1.0590966E-02   3.6895816E-01  -3.4132485E-01   4.2819085E-02
 -9.5089721E-02  -3.4132485E-01   2.9211059E+00  -1.3498471E-01
  5.6173730E-02   4.2819085E-02  -1.3498471E-01   1.0635575E+00

     F     MATRIX         2 ROWS        4 COLUMNS
  2.7657460E-01   4.4589747E-01  -1.5678530E+00   6.7515437E-01
  8.2414674E-02  -9.4596876E-01   4.6313338E-01   1.3877541E-01
```

```
RESIDUAL ERROR IN RICCATI EQUATION


     EROR  MATRIX        4 ROWS        4 COLUMNS
 -3.1086245E-15  -4.2188475E-15   1.1990409E-14  -4.6185278E-14
 -4.2743586E-15  -7.1054274E-15   1.7763568E-14  -7.5495166E-14
  1.1990409E-14   1.7763568E-14  -4.2632564E-14   1.9895197E-13
 -4.6851412E-14  -7.5051076E-14   1.9895197E-13  -8.3133500E-13

EIGENVALUES OF P


     EVLP  MATRIX        4 ROWS        1 COLUMNS
  4.5717003E-02
  3.2330141E-01
  1.0566117E+00
  2.9795117E+00

CLOSED-LOOP RESPONSE MATRIX A-BF


     A-BF  MATRIX        4 ROWS        4 COLUMNS
  3.5157081E-01   4.3479404E-03  -3.0473224E-01  -5.8984369E-01
  6.7402744E-03   8.0039079E-01   3.4384439E-01  -4.0426071E-03
  2.0223236E-03  -4.2003004E-02   9.7874931E-01   1.1903116E-03
  3.3465964E-02   3.3020260E-03  -7.2966874E-03   9.8493537E-01

EIGENVALUES OF A-BF


  3.8573552E-01   0.
  8.9933093E-01   7.9976165E-02
  8.8933093E-01  -7.9976185E-02
  9.5124891E-01   0.
```

```
EXAMPLE 2 - OPTIMAL SAMPLED-DATA REGULATOR                                                    ORACLS  PROGRAM

FOR THE ORIGINAL SAMPLED-DATA PROBLEM

  FHAT  MATRIX         2 ROWS          4 COLUMNS
2.7657460E-01   4.4589747E-01  -1.5678530E+00   6.7515437E-01
8.2414674E-02  -9.4596876E-01   4.6313338E-01   1.3877541E-01

  F     MATRIX         2 ROWS          4 COLUMNS
5.7401812E-01   4.7735433E-01  -1.9518615E+00   6.8027022E-01
2.0345396E-01  -1.0192663E+00   2.9723921E-01   1.4074207E-01
```

Example 3 - Model Following

This problem demonstrates the model-following capability of ORACLS and illustrates the output format of the transient response subroutine TRANSIT.

Given the linear system

$$\dot{x}(t) = A\,x(t) + B\,u(t)$$

with $x(0) = x_0$ given and A and B as stated in example 1, use the explicit model-following procedure to find a control law which causes the state $x(t)$ to track the state $\tilde{x}_m(t)$ of the system,

$$\dot{\tilde{x}}_m(t) = \tilde{A}_m\,\tilde{x}_m(t) + \tilde{B}_m\,\tilde{u}_m(t)$$

where $\tilde{x}_m(0) = \tilde{x}_m^o$ is given and the step input is

$$\tilde{u}_m(t) = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

with

$$\tilde{A}_m = \begin{bmatrix} -0.981 & 0.177 & -10.0 & 0.0 \\ 0.030 & -0.092 & 5.23 & 0.0 \\ 0.0 & -1.0 & -0.732 & 0.052 \\ 1.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

$$\tilde{B}_m = \begin{bmatrix} 6.34 & 4.58 \\ 0.690 & -2.65 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \end{bmatrix}$$

Generally, the state and model dynamics represent two sets of linearized equations and $u(t)$ is to cause the system represented by $x(t)$ to perform like the model $\tilde{x}_m(t)$ when the model is subjected to a particular step input. Data for the model dynamics are from handling-quality model 3 of reference 31.

The solution to the posed problem can be found by using the subroutine EXPMDFL of ORACLS. The foregoing problem can be placed in the format of EXPMDFL by redefining the model dynamics as

$$\dot{x}_m(t) = A_m\, x_m(t)$$

with

$$x_m(0) = \begin{bmatrix} \tilde{x}_m^o \\ \\ 2 \\ 0 \end{bmatrix}$$

and

$$y_m(t) = H_m\, x_m(t)$$

where

$$x_m(t) = \begin{bmatrix} \tilde{x}_m(t) \\ \tilde{u}_m(t) \end{bmatrix}$$

$$A_m = \left[\begin{array}{c:c} \tilde{A}_m & \tilde{B}_m \\ \hdashline 0_{2\times 4} & 0_{2\times 2} \end{array}\right]$$

and

$$H_m = \left[\begin{array}{c:c} I_4 & 0_{4\times 2} \end{array}\right]$$

where I_4 is the 4×4 identity matrix. Weighting matrices in EXPMDFL are chosen to be

$$Q = \text{diag}\ (10.,10.,10.,10.,)$$

and R is the 2×2 identity matrix. A transient response is evaluated using TRANSIT with

$$x(0) = \tilde{x}_m^0 = 0$$

The executive program and output data follow.

CONTENTS

SUMMARY 1 1/A9

INTRODUCTION 1 1/A9

OVERVIEW OF ORACLS 2 1/A10

PRIMARY SUBROUTINES FOR INPUT-OUTPUT 5 1/A13
  Subroutine RDTITL[1] 5 1/A13
    (Input Hollerith data; define COMMON block data)
  Subroutine LNCNT 6 1/A14
    (Accumulate line count; page output)
  Subroutine READ 7 1/B1
    (Input numerical data)
  Subroutine PRNT 8 1/B2
    (Output numerical data)

PRIMARY SUBROUTINES FOR VECTOR-MATRIX OPERATIONS 10 1/B4
  Subroutine EQUATE 10 1/B4
    (Equating matrices, A = B)
  Subroutine TRANP 11 1/B5
    (Matrix transpose, A')
  Subroutine SCALE 12 1/B6
    (Scalar multiplication)
  Subroutine UNITY 13 1/B7
    (Generate identity matrix)
  Subroutine NULL 14 1/B8
    (Generate null matrix)
  Subroutine TRCE 15 1/B9
    (Trace of matrix)
  Subroutine ADD 16 1/B10
    (Matrix addition)
  Subroutine SUBT 17 1/B11
    (Matrix subtraction)
  Subroutine MULT 18 1/B12
    (Matrix multiplication)
  Subroutine MAXEL 19 1/B13
    (Maximum of absolute values of matrix elements)
  Subroutine NORMS 20 1/B14
    (Compute ℓ_1, ℓ_2, and ℓ_∞ matrix norms)
  Subroutine JUXTC 22 1/C2
    (Matrix juxtaposition by columns)
  Subroutine JUXTR 23 1/C3
    (Matrix juxtaposition by rows)

---

[1] Required of all user-provided driver programs.

PRIMARY SUBROUTINES FOR ANALYSIS OF CONSTANT LINEAR SYSTEMS 24 1/C4
Subroutine FACTOR 24 1/C4
(Factor nonnegative definite matrix)
Subroutine EIGEN 26 1/C6
(Eigenvalue-eigenvector computation)
Subroutine SYMPDS 28 1/C8
(Solve $AX = B$, $A = A' > 0$)
Subroutine GELIM 30 1/C10
(Solve $AX = B$, A nonsingular)
Subroutine SNVDEC 32 1/C12
(Singular value decomposition)
Subroutine SUM 35 1/D1
(Solve discrete Liapunov equation)
Subroutine BILIN 37 1/D3
(Solve continuous Liapunov equation)
Subroutine BARSTW 41 1/D7
(Solve general equation $AX + XB = C$)
Subroutine TESTSTA 43 1/D9
(Test matrix for relative stability)
Subroutine EXPSER 45 1/D11
(Matrix exponential by series method)
Subroutine EXPADE 47 1/D13
(Matrix exponential by Padé method)
Subroutine EXPINT 49 1/E1
(Matrix exponential and integral)
Subroutine VARANCE 51 1/E3
(Steady-state variance of linear system)
Subroutine CTROL 54 1/E6
(Controllability matrix)
Subroutine TRANSIT 57 1/E9
(Transient response of forced linear system)

PRIMARY SUBROUTINES FOR IMPLEMENTING LQG CONTROL LAW DESIGN 60 1/E12
Subroutine SAMPL 60 1/E12
(Evaluate sampled data regulator coefficients)
Subroutine PREFIL 64 1/F2
(Eliminate performance index cross-product terms)
Subroutine CSTAB 67 1/F5
(Stabilize continuous systems)
Subroutine DSTAB 70 1/F8
(Stabilize discrete systems)
Subroutine DISCREG 74 1/F12
(Digital transient quadratic regulator)
Subroutine CNTNREG 78 1/G2
(Continuous transient quadratic regulator)
Subroutine RICTNWT 84 1/G8
(Riccati solution by Newton's method)
Subroutine ASYMREG 88 1/G12
(Asymptotic quadratic regulator)
Subroutine ASYMFIL 94 2/A7
(Asymptotic Kalman-Bucy filter)

Subroutine EXPMDFL 100 2/A13
(Explicit model following)
Subroutine IMPMDFL 108 2/B7
(Implicit model following)

SUPPORTING SUBROUTINES 115 2/B14
Subroutine READ1 . . . . 115 2/B14
(Read single matrix)
Subroutine BALANC 116 2/C1
(Supports eigenvalue/eigenvector computation)
Subroutine ELMHES 117 2/C2
(Supports eigenvalue/eigenvector computation)
Subroutine HQR 118 2/C3
(Supports eigenvalue/eigenvector computation)
Subroutine INVIT 120 2/C5
(Supports eigenvalue/eigenvector computation)
Subroutine ELMBAK 122 2/C7
(Supports eigenvalue/eigenvector computation)
Subroutine BALBAK 123 2/C8
(Supports eigenvalue/eigenvector computation)
Subroutine DETFAC 124 2/C9
(LU factorization of real square matrix)
Subroutine AXPXB 126 2/C11
(Solve AX + XB = C)
Subroutine SHRSLV 129 2/C14
(Supports BARSTW)
Subroutine ATXPXA 130 2/D1
(Solve A'X + XA = C, C = C')
Subroutine SYMSLV 132 2/D3
(Supports BARSTW)
Subroutine HSHLDR 133 2/D4
(Supports BARSTW)
Subroutine BCKMLT 134 2/D5
(Supports BARSTW)
Subroutine SCHUR 135 2/D6
(Supports BARSTW)
Subroutine SYSSLV 137 2/D8
(Supports BARSTW)
Subroutine GAUSEL 138 2/D9
(Supports EXPADE)

EXAMPLE COMPUTATIONS 139 2/D10
Example 1 - Optimal Transient Regulator 140 2/D11
Example 2 - Optimal Sampled-Data Regulator 152 2/F5
Example 3 - Model Following 163 2/G11
Example 4 - Kalman-Bucy Filter 183 3/C11

CONCLUDING REMARKS 190 3/D9

REFERENCES 191 3/D10

```
      PROGRAM  MODFOL(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C
      DIMENSION A(16),B(8),AM(36),HM(24),Q(16),R(4),F(20),P(40),DUMMY(70
     10),AC(100),BC(20),X(10)
      DIMENSION NA(2),NB(2),NAM(2),NHM(2),NQ(2),NR(2),NF(2),NP(2),NAC(2)
     1,NBC(2),IOP(5),NDUM(2),T(2),NX(2)
      LOGICAL  DISC,NEWT,STABLE,HIDENT,HMIDENT
C
C
C     INPUT HOLLERITH DATA FOR TITLE OF OUTPUT
      CALL RDTITL
C
C     INPUT COEFFICIENT MATRICES FOR PLANT AND MODEL
      CALL READ(5,A,NA,B,NB,AM,NAM,HM,NHM,X,NX)
C
C     DEFINE WEIGHTING MATRICES
      NQ(1)= NA(1)
      NQ(2)= NQ(1)
      NR(1)= NB(2)
      NR(2)= NR(1)
      CALL UNITY(Q,NQ)
      CALL SCALE(Q,NQ,Q,NQ,10.)
      CALL UNITY(R,NR)
C
C
C     COMPUTE MODEL-FOLLOWING GAINS
      IOP(1) = 1
      IOP(2) = 1
      IOP(3)=0
      IOP(4) = 0
      IOP(5) = 1
      DISC = .FALSE.
      NEWT = .FALSE.
      STABLE = .FALSE.
      HIDENT=.TRUE.
      HMIDENT = .FALSE.
C
      CALL EXPMDFL(A,NA,B,NB,H,NH,AM,NAM,HM,NHM,Q,NQ,R,NR,F,NF,P,NP,HIDE
     1NT,HMIDENT,DISC,NEWT,STABLE,FNULL,ALPHA,IOP,DUMMY)
C
C
C     GENERATE COEFFICIENT MATRICES FOR INPUT TO TRANSIENT
```

```
C     RESPONSE SUBROUTINE
      NDUM(1) = NAM(1)
      NDUM(2) = NA(1)
      CALL NULL(DUMMY,NDUM)
      CALL JUXTR(A,NA,DUMMY,NDUM,AC,NAC)
      NDUM(1) = NA(1)
      NDUM(2) = NAM(1)
      CALL NULL(DUMMY,NDUM)
      N1 = NDUM(1)*NDUM(2) + 1
      CALL JUXTR(DUMMY,NDUM,AM,NAM,DUMMY(N1),NHM)
      N2=NHM(1)*NHM(2)+N1
      CALL JUXTC(AC,NAC,DUMMY(N1),NHM,DUMMY(N2),NDUM)
      CALL EQUATE(DUMMY(N2),NDUM,AC,NAC)
      NDUM(1) = NAM(1)
      NDUM(2) = NB(2)
      CALL NULL(DUMMY,NDUM)
      CALL JUXTR(B,NB,DUMMY,NDUM,BC,NBC)
C
C     COMPUTE TRANSIENT RESPONSE TO OBSERVE MODEL-FOLLOWING ACCURACY
      IOP(1) = 0
      IOP(2) = 0
      IOP(3) = 0
      T(1) = 50.
      T(2) = 2.
C
      CALL LNCNT(100)
      PRINT 100,NA(1),NAM(1)
  100 FORMAT(*       IN THE TRAJECTORY OUTPUT TO FOLLOW THE FIRST*I4* CO
     1LUMNS CORRESPOND TO X TRANSPOSE*/*   AND THE NEXT*I4* COLUMNS TO X
     2M TRANSPOSE*)
C
      CALL TRANSIT(AC,NAC,BC,NBC,H,NH,G,NG,F,NF,V,NV,T,X,NX,DISC,STABLE,
     1IOP,DUMMY)
C
C
      STOP
      END
```

```
    EXAMPLE  3 - MODEL-FOLLOWING                                                       ORACLS  PROGRAM


        A     MATRIX          4 ROWS          4 COLUMNS
-2.6000000E+00   2.5000000E-01  -3.8000000E+01   0.
-7.5000000E-02  -2.7000000E-01   4.4000000E+00   0.
 7.8000000E-02  -9.9000000E-01  -2.3000000E-01   5.2000000E-02
 1.0000000E+00   7.9000000E-02   0.              0.

        B     MATRIX          4 ROWS          2 COLUMNS
 1.7000000E+01   7.0000000E+00
 8.2000000E-01  -3.2000000E+00
 0.              4.6000000E-02
 0.              0.

        AM    MATRIX          6 ROWS          6 COLUMNS
-9.8100000E-01   1.7700000E-01  -1.0000000E+01   0.               6.3400000E+00   4.5800000E+00
 3.0000000E-02  -9.2000000E-02   5.2300000E+00   0.              -6.9000000E-01  -2.6500000E+00
 0.             -1.0000000E+00  -7.3200000E-01   5.2000000E-02   0.              0.
 1.0000000E+00   0.              0.              0.              0.              0.
 0.              0.              0.              0.              0.              0.
 0.              0.              0.              0.              0.              0.

        HM    MATRIX          4 ROWS          6 COLUMNS
 1.0000000E+00   0.              0.              0.              0.              0.
 0.              1.0000000E+00   0.              0.              0.              0.
 0.              0.              1.0000000E+00   0.              0.              0.
 0.              0.              0.              1.0000000E+00   0.              0.

        X     MATRIX         10 ROWS          1 COLUMNS
 0.
 0.
 0.
 0.
 0.
 0.
 0.
 0.
 2.0000000E+00
 0.
```

```
PROGRAM TO SOLVE ASYMPTOTIC CONTINUOUS EXPLICIT MODEL-FOLLOWING PROBLEM

PLANT DYNAMICS


      A    MATRIX         4 ROWS        4 COLUMNS
 -2.6000000E+00   2.5000000E-01  -3.8000000E+01   0.
 -7.5000000E-02  -2.7000000E-01   4.4000000E+00   0.
  7.8000000E-02  -9.9000000E-01  -2.3000000E-01   5.2000000E-02
  1.0000000E+00   7.8000000E-02   0.              0.

      B    MATRIX         4 ROWS        2 COLUMNS
  1.7000000E+01   7.0000000E+00
  8.2000000E-01  -3.2000000E+00
  0.              4.6000000E-02
  0.              0.

H IS AN IDENTITY MATRIX


MODEL DYNAMICS


      AM   MATRIX         6 ROWS        6 COLUMNS
 -9.8100000E-01   1.7700000E-01  -1.0000000E+01   0.              6.3400000E+00   4.5800000E+00
  3.0000000E-02  -9.2000000E-02   5.2300000E+00   0.             -6.9000000E-01  -2.6500000E+00
  0.             -1.0000000E+00  -7.3200000E-01   5.2000000E-02   0.              0.
  1.0000000E+00   0.              0.              0.              0.              0.
  0.              0.              0.              0.              0.              0.
  0.              0.              0.              0.              0.              0.

      HM   MATRIX         4 ROWS        6 COLUMNS
  1.0000000E+00   0.              0.              0.              0.              0.
  0.              1.0000000E+00   0.              0.              0.              0.
  0.              0.              1.0000000E+00   0.              0.              0.
  0.              0.              0.              1.0000000E+00   0.              0.
```

170

```
WEIGHTING MATRICES


     Q    MATRIX         4 ROWS         4 COLUMNS
  1.0000000E+01   0.              0.              0.
  0.              1.0000000E+01   0.              0.
  0.              0.              1.0000000E+01   0.
  0.              0.              0.              1.0000000E+01

     R    MATRIX         2 ROWS         2 COLUMNS
  1.0000000E+00   0.
  0.              1.0000000E+00


CONTROL LAW U = -F( COL.(X,XM) ),  F = (F11,F12)


PART OF F MULTIPLYING  X


     F11  MATRIX         2 ROWS         4 COLUMNS
  2.8715216E+00   1.1867788E+00  -2.3493987E+00   3.0210811E+00
  1.0990931E+00  -3.0362141E+00   1.5921904E+00   9.3351305E-01

     P11  MATRIX         4 ROWS         4 COLUMNS
  1.6786659E-01   2.2109123E-02  -1.1051965E-01   1.7350075E-01
  2.2109123E-02   9.8893140E-01  -5.7385910E-01   8.8254034E-02
 -1.1051966E-01  -5.7385910E-01   1.1510412E+01   3.0884697E-02
  1.7350075E-01   8.8254034E-02   3.0894697E-02   1.0164474E+01

EIGENVALUES OF P11


  1.6346846E-01
  9.5710942E-01
  1.0167942E+01
  1.1543144E+01

PLANT CLOSED-LOOP RESPONSE MATRIX A - BF11
```

```
 -5.9109518E+01   1.3282587E+00  -9.2055556E+00  -5.7906570E+01
  1.0874501E+00  -1.0959044E+01   1.1421516E+01   5.0929927E-01
  2.7441719E-02  -8.5033415E-01  -3.0324076E-01   9.0583996E-03
  1.0000000E+00   7.8000000E-02   0.              0.

EIGENVALUES OF CLOSED-LOOP RESPONSE MATRIX


 -1.0076624E+00   0.
 -1.2943964E+00   0.
 -9.9317854E+00   0.
 -5.8137959E+01   0.

PART OF F MULTIPLYING XM


      F12  MATRIX        2 ROWS        6 COLUMNS
 -2.9990036E+00  -1.0535303E+00   9.7142903E-01  -3.0178671E+00  -5.5004933E-01  -4.3236218E-01
 -9.5061040E-01   2.9377885E+00  -1.5695172E+00  -9.2056907E-01   8.1556250E-01   4.4887921E-01

      P12  MATRIX        4 ROWS        6 COLUMNS
 -1.7290904E-01  -1.6058975E-02   3.6971673E-02  -1.7312936E-01  -2.0297024E-02  -1.9460473E-02
 -7.2621845E-02  -9.5186308E-01   4.1818364E-01  -9.1058595E-02  -2.4999989E-01  -1.2382211E-01
  5.9484706E-01   9.2161895E-02  -1.0655026E+01  -1.1101935E-03   3.4270003E+00   4.1059083E+00
 -1.5393360E-01   1.4943541E-01   4.0497750E-01  -1.0136463E+01   6.3234900E-02  -3.7895492E-01
```

```
        EXAMPLE  3 - MODEL-FOLLOWING                                                      ORACLS  PROGRAM

        IN THE TRAJECTORY OUTPUT TO FOLLOW THE FIRST   4 COLUMNS CORRESPOND TO X TRANSPOSE
     AND THE NEXT   6 COLUMNS TO XM TRANSPOSE


 COMPUTATION OF TRANSIENT RESPONSE FOR THE CONTINUOUS   SYSTEM


        A    MATRIX       10 ROWS      10 COLUMNS
  -2.6000000E+00   2.5000000E-01  -3.8000000E+01   0.             0.             0.             0.
      0.             0.             0.
  -7.5000000E-02  -2.7000000E-01   4.4000000E+00   0.             0.             0.             0.
      0.             0.             0.
   7.8000000E-02  -9.9000000E-01  -2.3000000E-01   5.2000000E-02  0.             0.             0.
      0.             0.             0.
   1.0000000E+00   7.8000000E-02   0.             0.             0.             0.             0.
      0.             0.             0.
   0.             0.             0.             0.            -9.8100000E-01  1.7700000E-01 -1.0000000E+01
      0.             6.3400000E+00  4.5800000E+00
   0.             0.             0.             0.             3.0000000E-02 -9.2000000E-02  5.2300000E+00
      0.            -6.9000000E-01 -2.6500000E+00
   0.             0.             0.             0.             0.            -1.0000000E+00 -7.3200000E-01
      5.2000000E-02  0.             0.
   0.             0.             0.             0.             1.0000000E+00  0.             0.
      0.             0.             0.
   0.             0.             0.             0.             0.             0.             0.
      0.             0.             0.
   0.             0.             0.             0.             0.             0.             0.
      0.             0.             0.

        B    MATRIX       10 ROWS       2 COLUMNS
   1.7000000E+01   7.0000000E+00
   8.2000000E-01  -3.2000000E+00
   0.              4.6000000E-02
   0.              0.
   0.              0.
   0.              0.
   0.              0.
   0.              0.
   0.              0.
   0.              0.

 H IS A NULL MATRIX
```

```
G IS A NULL MATRIX

      F     MATRIX        2 ROWS       10 COLUMNS
  2.8715216E+00   1.1967788E+00  -2.3493987E+00   3.0218811E+00  -2.9990036E+00  -1.0535303E+00   9.7142903E-01
    -3.0178671E+00  -5.5004933E-01  -4.3236218E-01
  1.0990931E+00  -3.0362141E+00   1.5921904E+00   9.3351305E-01  -9.5061040E-01   2.9377885E+00  -1.5695172E+00
    -9.2056907E-01   8.1556250E-01   4.4887921E-01

V IS A NULL MATRIX

COMPUTATION OF THE MATRIX EXPONENTIAL EXP(A T) BY THE SERIES METHOD

      A     MATRIX       10 ROWS       10 COLUMNS
 -5.9109518E+01   1.3282587E+00  -9.2055556E+00  -5.7906570E+01   5.7637333E+01  -2.6545044E+00  -5.5276735E+00
     5.7747725E+01   3.6419010E+00   4.2080025E+00
  1.0874501E+00  -1.0959044E+01   1.1421516E+01   5.0929927E-01  -5.8277036E-01   1.0264818E+01  -5.8190267E+00
    -4.7116998E-01   3.0608404E+00   1.7909505E+00
  2.7441719E-02  -8.5033415E-01  -3.0324076E-01   9.0583996E-03   4.3728078E-02  -1.3513827E-01   7.2197789E-02
     4.2346177E-02  -3.7515875E-02  -2.0648444E-02
  1.0000000E+00   7.8000000E-02   0.              0.              0.              0.              0.
     0.              0.              0.
  0.              0.              0.              0.             -9.8100000E-01   1.7700000E-01  -1.0000000E+01
     0.              6.3400000E+00   4.5800000E+00
  0.              0.              0.              0.              3.0000000E-02  -9.2000000E-02   5.2300000E+00
     0.             -6.9000000E-01  -2.6500000E+00
  0.              0.              0.              0.              0.             -1.0000000E+00  -7.3200000E-01
     5.2000000E-02   0.              0.
  0.              0.              0.              0.              1.0000000E+00   0.              0.
     0.              0.              0.
  0.              0.              0.              0.              0.              0.              0.
     0.              0.              0.
  0.              0.              0.              0.              0.              0.              0.
     0.              0.              0.

T =     .20000000E+01
```

```
     EXAMPLE  3 - MODEL-FOLLOWING                                                                          ORACLS  PROGRAM

     EXPA  MATRIX        10 ROWS        10 COLUMNS
 -2.3891131E-03  -1.3251850E-03  -1.1385082E-04  -1.3738924E-01    1.1955605E-01   1.0625337E-01   2.1427525E+00
    1.2605585E-01   3.9547601E+00  -1.4409641E+00
 -1.3074054E-04  -8.5670545E-03   9.7002193E-02  -8.5196158E-04    6.2644729E-02   1.4534787E-02  -1.0599349E+00
    5.3247219E-02   7.2397266E-01   3.8807667E-01
  1.4763163E-05  -7.2188507E-03   8.2534175E-02   6.5207081E-03    9.2176534E-03   2.2259046E-01  -1.5515221E-01
   -1.6411698E-02   8.2364845E-02   4.8831924E-01
  2.3921962E-03   2.2911988E-03  -1.0898440E-02   1.3680976E-01    8.4821386E-01   2.0037391E+00  -1.2295579E-01
    7.6035997E-01   5.8562550E+00   2.3683158E-01
  0.              0.              0.              0.               1.2107298E-01   1.1670875E-01   2.1381511E+00
   -7.8748841E-03   4.0115443E+0C  -1.4422067E+00
  0.              0.              0.              0.               4.8006911E-02  -3.5454142E-02  -1.0445038E+00
    5.4105409E-02   3.0701214E-01   1.1717037E-01
  0.              0.              0.              0.               1.2870845E-03   2.0066917E-01  -1.1693378E-02
   -9.3881938E-03   1.7719426E-01   5.7703078E-01
  0.              0.              0.              0.               8.5227516E-01   2.0172177E+00  -1.5144008E-01
    9.9663838E-01   5.8926293E+00   2.6304712E-01
  0.              0.              0.              0.               0.              0.              0.
    0.              1.0000000E+00   0.
  0.              0.              0.              0.               0.              0.              0.
    0.              0.              1.0000000E+00

 STRUCTURE OF PRINTING TO FOLLOW

   TIME OR STAGE
  STATE - X TRANSPOSE - FROM DX = AX + BU
  OUTPUT - Y TRANSPOSE - FROM Y = HX + GU   IF DIFFERENT FROM X
  CONTROL - U TRANSPOSE - FROM U = -FX + V



  0.


  0.              0.              0.              0.               0.              0.              0.
    0.              2.0000000E+00   0.
```

```
1.1000987E+00  -1.6311250E+00



 .2000000E+01

7.9095202E+00   1.4479453E+00   1.6472969E-01   1.1712510E+01   8.0230886E+00   6.1402428E-01   3.5438851E-01
    1.1785259E+01   2.0000000E+00   0.

1.5927996E+00   1.0410762E-01



 .4000000E+01

9.5489317E+00   2.2039396E+00   2.0659338E-01   3.0288443E+01   9.7310584E+00   1.2449044E+00   3.7314436E-01
    3.0375205E+01   2.0000000E+00   0.

1.8227148E+00   1.0305996E-01



 .6000000E+01

9.6466669E+00   3.2716314E+00   1.7391236E-01   4.9680598E+01   9.9051858E+00   2.2907579E+00   3.2719551E-01
    4.9769107E+01   2.0000000E+00   0.

1.7942873E+00   6.0926728E-02
```

```
 .8000000E+01

9.4589500E+00   4.3502637E+00   2.1320376E-01   6.9331980E+01   9.7973579E+00   3.3593457E+00   3.5575370E-01
   6.9423499E+01   2.0000000E+00   0.

1.8504096E+00   3.1082298E-02


 .1000000E+02

9.3974701E+00   5.3531655E+00   2.4666911E-01   8.9012689E+01   9.8153021E+00   4.3498632E+00   3.8519443E-01
   8.9105733E+01   2.0000000E+00   0.

1.9097142E+00  -9.6113310E-03


 .1200000E+02

9.3439028E+00   6.3534006E+00   2.6358106E-01   1.0866891E+02   9.8410301E+00   5.3497710E+00   3.9889896E-01
   1.0876250E+02   2.0000000E+00   0.

1.9444176E+00  -5.5910245E-02


 .1400000E+02
```

```
9.2585870E+00   7.3479771E+00   2.8394166E-01   1.2832999E+02   9.0352651E+00   6.3648203E+00   4.1484140E-01
   1.2842441E+02   2.0000000E+00   0.

1.9814221E+00  -9.9857508E-02



 .1600000E+02

9.1761171E+00   8.4098289E+00   3.0715116E-01   1.4799891E+02   9.0323699E+00   7.3756775E+00   4.3374640E-01
   1.4809427E+02   2.0000000E+00   0.

2.0230646E+00  -1.4313242E-01



 .1800000E+02

9.0997321E+00   9.4274267E+00   3.2918291E-01   1.6766850E+02   9.0355191E+00   8.3841989E+00   4.5170504E-01
   1.6776481E+02   2.0000000E+00   0.

2.0634769E+00  -1.8702727E-01



 .2000000E+02

9.0217633E+00   1.0447312E+01   3.5081025E-01   1.8733946E+02   9.0370988E+00   9.3941185E+00   4.6920740E-01
   1.8743654E+02   2.0000000E+00   0.
```

```
2.1031212E+00  -2.3097232E-01



 .2200000E+02

8.9427645E+00   1.1467616E+01   3.7275076E-01   2.0701295E+02   9.8376666E+00   1.0404454E+01   4.8698248E-01
   2.0711089E+02   2.0000000E+00   0.

2.1431502E+00  -2.7478692E-01



 .2400000E+02

8.8642480E+00   1.2487737E+01   3.9472902E-01   2.2668872E+02   9.8387229E+00   1.1414584E+01   5.0481197E-01
   2.2678753E+02   2.0000000E+00   0.

2.1832840E+00  -3.1861843E-01



 .2600000E+02

8.7859578E+00   1.3507953E+01   4.1663924E-01   2.4636654E+02   9.8399126E+00   1.2424811E+01   5.2257858E-01
   2.4646621E+02   2.0000000E+00   0.

2.2233270E+00  -3.6248104E-01



 .2800000E+02
```

```
8.7073351E+00   1.4528345E+01   4.3855571E-01   2.6604650E+02   9.8409794E+00   1.3435217E+01   5.4034652E-01
   2.6614705E+02   2.0000000E+00   0.

2.2633685E+00  -4.0634175E-01



 .3000000E+02

8.6287953E+00   1.5548843E+01   4.6048866E-01   2.8572865E+02   9.8420381E+00   1.4445726E+01   5.5813003E-01
   2.9583006E+02   2.0000000E+00   0.

2.3034352E+00  -4.5020286E-01



 .3200000E+02

8.5502742E+00   1.6569440E+01   4.8242098E-01   3.0541296E+02   9.8431238E+00   1.5456333E+01   5.7591345E-01
   3.0551524E+02   2.0000000E+00   0.

2.3435043E+00  -4.9407089E-01



 .3400000E+02

8.4717418E+00   1.7590152E+01   5.0435325E-01   3.2509942E+02   9.8442071E+00   1.6467054E+01   5.9369631E-01
   3.2520257E+02   2.0000000E+00   0.
```

```
2.3835737E+00  -5.3794431E-01



 .3600000E+02

8.3931956E+00   1.8610979E+01   5.2628895E-01   3.4478805E+02   9.8452953E+00   1.7477889E+01   6.1148195E-01
   3.4489206E+02   2.0000000E+00   0.

2.4236486E+00  -5.8182202E-01



 .3800000E+02

8.3146421E+00   1.9631916E+01   5.4822741E-01   3.6447983E+02   9.8463650E+00   1.8488833E+01   6.2926992E-01
   3.6458371E+02   2.0000000E+00   0.

2.4637285E+00  -6.2570450E-01



 .4000000E+02

8.2360809E+00   2.0652965E+01   5.7016800E-01   3.8417177E+02   9.8474457E+00   1.9499888E+01   6.4705962E-01
   3.8427752E+02   2.0000000E+00   0.

2.5038125E+00  -6.6959191E-01



 .4200000E+02
```

```
8.1575106E+00   2.1674127E+01   5.9211098E-01   4.0386688E+02   9.8485261E+00   2.0511054E+01   6.6485122E-01
   4.0397349E+02   2.0000000E+00   0.

2.5439009E+00  -7.1348413E-01



 .4400000E+02

8.0789316E+00   2.2695400E+01   6.1405642E-01   4.2356414E+02   9.8496065E+00   2.1522331E+01   6.8264483E-01
   4.2367162E+02   2.0000000E+00   0.

2.5839937E+00  -7.5738114E-01



 .4600000E+02

8.0003441E+00   2.3716786E+01   6.3600426E-01   4.4326357E+02   9.8506871E+00   2.2533719E+01   7.0044040E-01
   4.4337192E+02   2.0000000E+00   0.

2.6240909E+00  -8.0128297E-01



 .4800000E+02

7.9217480E+00   2.4738283E+01   6.5795450E-01   4.6296516E+02   9.8517678E+00   2.3545218E+01   7.1823790E-01
   4.6307437E+02   2.0000000E+00   0.
```

```
    EXAMPLE  3 - MODEL-FOLLOWING                                                              ORACLS  PROGRAM

2.6641925E+00  -8.4518962E-01



 .5000000E+02

7.8431432E+00   2.5759893E+01   6.7990715E-01   4.8266891E+02   9.8528486E+00   2.4556828E+01   7.3603736E-01
   4.8277899E+02   2.0000000E+00   0.

2.7042985E+00  -8.9910109E-01
```

Example 4 - Kalman-Bucy Filter

This problem illustrates the asymptotic optimal estimator design capability of ORACLS. Consider the linear system

$$\dot{x}(t) = A\,x(t) + B\,u(t) + \xi(t)$$

with A and B from example 1 and $\xi(t)$ a zero-mean Gaussian white-noise process with constant intensity $Q = Q' \geq 0$. If the system is digitally controlled using a zero-order hold with sampling times equal to integral multiples $\Delta = 0.05$ second, the state equation becomes

$$x\left[(i+1)\Delta\right] = \tilde{A}\,x(i\Delta) + \tilde{B}\,u(i\Delta) + \tilde{\xi}(i\Delta) \qquad (i = 0,1,\ldots)$$

where

$$\tilde{A} = e^{A\Delta}$$

$$\tilde{B} = \int_0^\Delta e^{A\tau}B\,d\tau$$

and $\tilde{\xi}(i\Delta)$ is a zero-mean Guassian discrete white-noise process with variance matrix

$$\tilde{Q} = \int_0^\Delta e^{A\tau}Qe^{A'}\,d\tau$$

Observations are made at the sampling instants and are modeled by

$$y(i\Delta) = H\,x(i\Delta) + D\,u(i\Delta) + \eta(i\Delta)$$

where

$$H = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ -0.0852 & -0.0421 & -2.24 & -0.0021 \end{bmatrix}$$

$$D = \begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \\ 0.713 & -0.403 \end{bmatrix}$$

and $\eta(i\Delta)$ is a zero-mean Gaussian discrete white-noise process with variance $R = R' > 0$. Assuming $\tilde{\xi}(i\Delta)$ and $\eta(i\Delta)$ are mutually uncorrelated, find a Kalman-Bucy predictor to asymptotically estimate the state $x(i\Delta)$ from knowledge of the measurements $y(i\Delta)$ and control inputs $u(i\Delta)$ up to time $(i-1)\Delta$.

If the estimate of x is denoted by $\hat{x}$, the predictor has the structure (ref. 4)

$$\hat{x}\left[(i+1)\Delta\right] = \tilde{A}\, x(i\Delta) + \tilde{B}\, u(i\Delta) + F\left[y(i\Delta) - H\, \hat{x}(i\Delta) - D\, u(i\Delta)\right]$$

The filter gain F can be computed directly from ASYMFIL after ignoring the $\tilde{B}$ and D matrices in the digital plant and observation equations. As in example 2, $\tilde{A}$ and $\tilde{B}$ follow from EXPINT. The matrix $\tilde{Q}$ can be found from SAMPL using A' in place of A. Finally, we take Q and R to be 4×4 and 3×3 identity matrices, respectively.

The executive program and output data follow.

```
      PROGRAM KBFIL(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C
      DIMENSION A(16),B(8),H(12),Q(16),R(9),ATIL(16),BTIL(8),QTIL(16),F(
     112),P(16),DUMMY(114)
      DIMENSION NA(2),NB(2),NH(2),NQ(2),NR(2),NATIL(2),NBTIL(2),NQTIL(2)
     1,IOP(5),NF(2),NP(2),NDUM(2)
      LOGICAL  IDENT,DISC,NEWT,STABLE,FNULL
C
C
C     INPUT HOLLERITH DATA FOR TITLE OF OUTPUT
      CALL RDTITL
C
C     INPUT COEFFICIENT MATRICES AND NOISE INTENSITIES
C     FOR CONTINUOUS SYSTEM
      CALL READ(5,A,NA,B,NB,H,NH,Q,NQ,R,NR)
C
C     GENERATE COEFFICIENT MATRICES FOR DIGITAL SYSTEM
      IOP(1)=0
      DELT= .05
      N1=(NA(1)**2) + 1
      CALL EXPINT(A,NA,ATIL,NATIL,DUMMY,NA,DELT,IOP,DUMMY(N1))
      CALL MULT(DUMMY,NA,B,NB,BTIL,NBTIL)
      CALL PRNT(ATIL,NATIL,4HATIL,1)
      CALL PRNT(BTIL,NBTIL,4HBTIL,1)
C
C     COMPUTE VARIANCE MATRIX FOR PROCESS NOISE OF DIGITAL SYSTEM
      CALL EQUATE(Q,NQ,QTIL,NQTIL)
      CALL TRANP(A,NA,DUMMY,NDUM)
      IOP(2)= 0
      CALL SAMPL(DUMMY,NDUM,B,NB,QTIL,NQTIL,R,NR,W,NW,DELT,IOP,DUMMY(N1)
     1)
      CALL PRNT(QTIL,NQTIL,4HQTIL,1)
C
C     SOLVE FOR DIGITAL FILTER GAIN
      IDENT = .TRUE.
      DISC  = .TRUE.
      STABLE = .FALSE.
      FNULL  =  .TRUE.
      NEWT   =  .TRUE.
      ALPHA=.5
      IOP(1)=1
      IOP(2)=0
      IDP(3)=0
      IDP(4)=0
      IDP(5)=0
      CALL ASYMFIL(ATIL,NATIL,G,NG,H,NH,QTIL,NQTIL,R,NR,F,NF,P,NP,IDENT,
     1DISC,NFWT,STABLE,FNULL,ALPHA,IOP,DUMMY)
C
C
      STOP
      END
     EXAMPLE  4   KALMAN-BUCY FILTER                                                       ORACLS  PROGRAM

     A     MATRIX        4 ROWS         4 COLUMNS
-2.6000000E+00   2.5000000E-01  -3.8000000E+01   0.
-7.5000000E-02  -2.7000000E-01   4.4000000E+00   0.
 7.8000000E-02  -9.9000000E-01  -2.3000000E-01   5.2000000E-02
 1.0000000E+00   7.8000000E-02   0.              0.

     B     MATRIX        4 ROWS         2 COLUMNS
 1.7000000E+01   7.0000000E+00
 8.2000000E-01  -3.2000000E+00
 0.              4.6000000E-02
 0.              0.

     H     MATRIX        3 ROWS         4 COLUMNS
 1.0000000E+00   0.              0.              0.
 0.              1.0000000E+00   0.              0.
-8.5200000E-02  -4.2100000E-02  -2.2400000E+00  -2.1000000E-03

     Q     MATRIX        4 ROWS         4 COLUMNS
 1.0000000E+00   0.              0.              0.
 0.              1.0000000E+00   0.              0.
 0.              0.              1.0000000E+00   0.
 0.              0.              0.              1.0000000E+00

     R     MATRIX        3 ROWS         3 COLUMNS
 1.0000000E+00   0.              0.
 0.              1.0000000E+00   0.
 0.              0.              1.0000000E+00

    ATIL   MATRIX        4 ROWS         4 COLUMNS
 8.7460216E-01   5.6206553E-02  -1.7644035E+00  -2.3524289E-03
-3.0699872E-03   9.8114690E-01   2.1998505E-01   2.8616468E-04
 3.7743137E-03  -4.8707923E-02   9.7958437E-01   2.5772852E-03
 4.6820759E-02   4.9177112E-03  -4.4809768E-02   9.9996068E-01

    BTIL   MATRIX        4 ROWS         2 COLUMNS
 7.9692392E-01   3.2234606E-01
 3.9250218E-02  -1.5895733E-01
 6.1800654E-04   6.8676137E-03
 2.0435169E-02   7.9852845E-03

    QTIL   MATRIX        4 ROWS         4 COLUMNS
```

```
  9.7864533E-02  -5.6349895E-03  -4.4643769E-02   2.0737669E-03
 -5.6349895E-03   4.9959106E-02   4.2255496E-03  -1.2169588E-05
 -4.4643769E-02   4.2255496E-03   4.9172184E-02  -6.8173577E-04
  2.0737669E-03  -1.2169588E-05  -6.8173577E-04   5.0057551E-02

PROGRAM TO SOLVE THE DISCRETE INFINITE-DURATION OPTIMAL FILTER PROBLEM


      A    MATRIX        4 ROWS        4 COLUMNS
  8.7460216E-01   5.6206553E-02  -1.7644035E+00  -2.3524288E-03
 -3.0698872E-03   9.5114690E-01   2.1998505E-01   2.8616469E-04
  3.7743137E-03  -4.8707923E-02   9.7958437E-01   2.5772852E-03
  4.6820759E-02   4.9177112E-03  -4.4809768E-02   9.9996068E-01

G IS AN IDENTITY MATRIX


      H    MATRIX        3 ROWS        4 COLUMNS
  1.0000000E+00   0.              0.              0.
  0.              1.0000000E+00   0.              0.
 -8.5200000E-02  -4.2100000E-02  -2.2400000E+00  -2.1000000E-03

INTENSITY MATRIX FOR COVARIANCE OF MEASUREMENT NOISE


      R    MATRIX        3 ROWS        3 COLUMNS
  1.0000000E+00   0.              0.
  0.              1.0000000E+00   0.
  0.              0.              1.0000000E+00

INTENSITY MATRIX FOR COVARIANCE OF PROCESS NOISE


      Q    MATRIX        4 ROWS        4 COLUMNS
  9.7864533E-02  -5.6349895E-03  -4.4643769E-02   2.0737669E-03
 -5.6349895E-03   4.9959106E-02   4.2255496E-03  -1.2169588E-05
 -4.4643769E-02   4.2255496E-03   4.9172184E-02  -6.8173577E-04
  2.0737669E-03  -1.2169588E-05  -6.8173577E-04   5.0057551E-02

FILTER GAIN
```

```
     EXAMPLE  4   KALMAN-BUCY FILTER                                                                    ORACLS  PROGRAM


         F     MATRIX          4 ROWS          3 COLUMNS
   5.3454233E-01  -1.0000879E-02   3.6312717E-01
  -4.3653081E-02   1.8601819E-01  -3.4324468E-02
  -8.4771306E-02  -9.0256464E-03  -1.2135706E-01
  -7.0927842E-03   3.8165991E-02  -1.4113759E-01

 STEADY-STATE VARIANCE MATRIX OF RECONSTRUCTION ERROR


         P     MATRIX          4 ROWS          4 COLUMNS
   8.8649680E-01  -5.7194386E-02  -2.2094025E-01  -1.2626090E-01
  -5.7194386E-02   2.3557573E-01   8.3030258E-03   5.2808810E-02
  -2.2094025E-01   8.3030258E-03   1.0595196E-01   9.1725042E-02
  -1.2626090E-01   5.2808810E-02   9.1725042E-02   1.9945248E+01

 EIGENVALUES OF P


        EVLP  MATRIX           4 ROWS          1 COLUMNS
   4.7257495E-02
   2.3111030E-01
   9.4813881E-01
   1.9946666E+01

        A-FH  MATRIX           4 ROWS          4 COLUMNS
   3.7099927E-01   8.1495386E-02  -9.5099854E-01  -1.5898617E-03
   3.7658749E-02   7.9368365E-01   1.4309824E-01   2.1408330E-04
   7.8205998E-02  -4.4791409E-02   7.0774455E-01   2.3224354E-03
   4.1888621E-02  -3.9190172E-02  -3.6095797E-01   9.9966429E-01

 EIGENVALUES OF A-FH MATRIX



   5.4024239E-01   2.2217840E-01
   5.4024239E-01  -2.2217840E-01
   7.9425006E-01   0.
   9.9735592E-01   0.
```

CONCLUDING REMARKS

This report has presented some 60 subroutines (43 primary purpose and 17 supporting) which can be used for the analysis and design of state-variable feedback control laws for time-invariant linear systems. The basic synthesis approach is through the well-founded linear-quadratic-Gaussian (LQG) methodology. The ORACLS system represents an attempt on the author's part to employ some of today's best numerical linear algebra techniques to the LQG design problem. A modular structure has been incorporated so that new methods can be incorporated, as they are developed.

The examples presented indicate only some of the capability of the ORACLS package. The program has been applied at the Langley Research Center to problems in aircraft design using state-variable equations of order up to 16. The Liapunov equations (subroutines BARSTW and BILIN) have been used successfully to solve equations of 79th order.

Each subroutine was compiled on a FORTRAN Extended compiler and executed on a Control Data Cyber digital computer system. The full ORACLS package (60 subroutines) requires 22 345 decimal (53 511 octal) words of storage.

Langley Research Center
National Aeronautics and Space Administration
Hampton, VA 23665
November 1, 1977

REFERENCES

1. Athans, Michael: The Role and Use of the Stochastic Linear-Quadratic-Gaussian Problem in Control System Design. IEEE Trans. Autom. Control, vol. AC-16, no. 6, Dec. 1971, pp. 529-552.

2. Wilkinson, J. H.; and Reinsch, C.: Handbook for Automatic Computation. Volume II - Linear Algebra. Springer-Verlag, 1971.

3. Kantorovich, L. V.; and Akilov, G. P. (D. E. Brown, trans.): Functional Analysis in Normed Spaces. A. P. Robertson, ed., MacMillan Co., 1964.

4. Kwakernaak, Huibert; and Sivan, Raphael: Linear Optimal Control Systems. John Wiley & Sons, Inc., c.1972.

5. Bartels, R. H.; and Stewart, G. W.: Algorithm 432 - Solution of the Matrix Equation $AX + XB = C$. Commun. ACM, vol. 15, no. 9, Sept. 1972, pp. 820-826.

6. Smith, R. A.: Matrix Equation $XA + BX = C$. SIAM J. Appl. Math., vol. 16, no. 2, Mar. 1968, pp. 198-201.

7. Luenberger, David G.: An Introduction to Observers. IEEE Trans. Autom. Control, vol. AC-16, no. 6, Dec. 1971, pp. 596-602.

8. Vaughan, David R.: A Negative Exponential Solution for the Matrix Riccati Equation. IEEE Trans. Autom. Control, vol. AC-14, no. 1, Feb. 1969, pp. 72-75.

9. Kleinman, David L.: On an Iterative Technique for Riccati Equation Computations. IEEE Trans. Autom. Control, vol. AC-13, no. 1, Feb. 1968, pp. 114-115.

10. Hewer, Gary A.: An Iterative Technique for the Computation of the Steady State Gains for the Discrete Optimal Regulator. IEEE Trans. Autom. Control, vol. AC-16, no. 4, Aug. 1971, pp. 382-383.

11. Armstrong, Ernest S.: An Extension of Bass' Algorithm for Stabilizing Linear Continuous Constant Systems. IEEE Trans. Autom. Control, vol. AC-20, no. 1, Feb. 1975, pp. 153-154.

12. Armstrong, Ernest S.; and Rublein, George T.: A Stabilization Algorithm for Linear Discrete Constant Systems. IEEE Trans. Autom. Control, vol. AC-21, no. 4, Aug. 1976, pp. 629-631.

13. Armstrong, Ernest S.; and Caglayan, Alper K.: An Algorithm for the Weighting Matrices in the Sampled-Data Optimal Linear Regulator Problem. NASA TN D-8372, 1976.

14. Tyler, J. S., Jr.: The Characteristics of Model-Following Systems as Synthesized by Optimal Control. IEEE Trans. Autom. Control, vol. AC-9, no. 4, Oct. 1964, pp. 485-498.

15. Armstrong, E. S.: Digital Explicit Model Following With Unstable Model Dynamics. AIAA Paper No. 74-888, AIAA Mechanics and Control of Flight Conference, Aug. 1974.

16. White, John S.; and Lee, Homer Q.: Users Manual for the Variable Dimension Automatic Synthesis Program (VASP). NASA TM X-2417, 1971.

17. Parlett, B. N.; and Reinsch, C.: Balancing a Matrix for Calculation of Eigenvalues and Eigenvectors. Numer. Math., Bd. 13, Heft 4, 1969, pp. 293-304.

18. Martin, R. S.; and Wilkinson, J. H.: Similarity Reduction of a General Matrix to Hessenberg Form. Numer. Math., Bd. 12, Heft 5, 1968, pp. 349-368.

19. Martin, R. S.; Peters, G.; and Wilkinson, J. H.: The QR Algorithm for Real Hessenberg Matrices. Numer. Math., Bd. 14, Heft 3, 1970, pp. 219-231.

20. Källström, Claes: Computing Exp (A) and $\int$ Exp (As) ds. Rep. 7309, Lund Inst. Technol. (Sweden), Mar. 1973.

21. Ward, R. C.: Numerical Computation of the Matrix Exponential With Accuracy Estimate. UCCND-CSD-24, Nov. 1975.

22. Desoer, C. A.: Notes for a Second Course on Linear Systems. Van Nostrand Reinhold Co., c.1970.

23. Dorato, Peter; and Levis, Alexander H.: Optimal Linear Regulators: The Discrete-Time Case. IEEE Trans. Autom. Control, vol. AC-16, no. 6, Dec. 1971, pp. 613-620.

24. Levis, Alexander H.; Schlueter, Robert A.; and Athans, Michael: On the Behaviour of Optimal Linear Sampled-Data Regulators. Int. J. Control, vol. 13, no. 2, Feb. 1971, pp. 343-361.

25. Armstrong, Ernest S.; and Rublein, George T.: A Discrete Analog of the Extended Bass Algorithm for Stabilizing Constant Linear Systems. Proceedings of the 1976 IEEE Conference on Decision and Control Including the 15th Symposium on Adaptive Processes, 76CH1150-2CS, Dec. 1976, pp. 1129-1131.

26. Åström, Karl J.: Introduction to Stochastic Control Theory. Academic Press, Inc., 1970.

27. Sandell, Nils R., Jr.: On Newton's Method for Riccati Equation Solution. IEEE Trans. Autom. Control, vol. AC-19, no. 3, June 1974, pp. 254-255.

28. Anderson, Brian D. O.; and Moore, John B.: Linear Optimal Control. Prentice-Hall, Inc., c.1971.

29. Wilkinson, J. H.: The Algebraic Eigenvalue Problem. Clarendon Press (Oxford), 1965.

30. Elliott, Jarrell R.: NASA's Advanced Control Law Program for the F-8 Digital Fly-by-Wire Aircraft. IEEE Trans. Autom. Control, vol. AC-22, no. 5, Oct. 1977, pp. 753-756.

31. Stein, Gunter; and Henke, Allen H.: A Design Procedure and Handling-Quality Criteria for Lateral-Directional Flight Control Systems. AFFDL-TR-70-152, U.S. Air Force, May 1971.

1. Report No. NASA TP-1106	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle ORACLS - A SYSTEM FOR LINEAR-QUADRATIC-GAUSSIAN CONTROL LAW DESIGN	5. Report Date April 1978 6. Performing Organization Code
7. Author(s) Ernest S. Armstrong	8. Performing Organization Report No. L-11769
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665	10. Work Unit No. 505-07-13-02 11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546	13. Type of Report and Period Covered Technical Paper 14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

A modern control theory design package (ORACLS) for constructing controllers and optimal filters for systems modeled by linear time-invariant differential or difference equations is described. The digital FORTRAN-coded ORACLS system represents an application of some of today's best numerical linear-algebra procedures to implement the linear-quadratic-Gaussian (LQG) methodology of modern control theory. Included are algorithms for computing eigensystems of real matrices, the relative stability of a matrix, factored forms for nonnegative definite matrices, the solutions and least squares approximations to the solutions of certain linear matrix algebraic equations, the controllability properties of a linear time-invariant system, and the steady-state covariance matrix of an open-loop stable system forced by white noise. These subroutines are applied to implement the various techniques of the LQG methodology. Subroutines are provided for solving both the continuous and discrete optimal linear regulator problems with noise-free measurements and the sampled-data optimal linear regulator problem. For measurement noise, duality theory and the optimal regulator algorithms are used to solve the continuous and discrete Kalman-Bucy filter problems. Subroutines are also included which give control laws causing the output of a system to track the output of a prescribed model. Finally, numerical examples are presented to illustrate the capability of the ORACLS system.

17. Key Words (Suggested by Author(s)) Matrix computation Linear system analysis Optimal control Liapunov and Riccati equations Kalman filtering Stochastic control	18. Distribution Statement Unclassified - Unlimited Subject Category 63

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 197	22. Price* $9.00

* For sale by the National Technical Information Service, Springfield, Virginia 22161